hard core

THE JOURNAL OF THE BRITISH APPLE SYSTEMS USER GROUP

MAY 1981 £1 VOLUME 1 Nº 3

BASUG at the LONDON COMPUTER FAIR

hardcore

THE JOURNAL OF
THE BRITISH APPLE SYSTEMS USER GROUP
P.O.BOX 174 WATFORD WD2 6NF

EDITED BY DAVID BOLTON

MEETINGS

Tuesday 12 May	7.30	Starting Machine Code
Sunday 24th May	2.30	Games
Tuesday 2 June	7.30	Open Evening
Sunday 21 June	2.30	ANNUAL GENERAL MEETING

Unless otherwise advised, these meetings will take place at The Old School, Branch Road, Park Street, St Albans. (On the old A5, about two miles south of St Albans town centre)

CONTENTS

EDITORIAL

BIGGER & BETTER

Well, here's the third issue of Hardcore, and the quality continues to improve - both in terms of presentation and of content. The most obvious change will be that, thanks to the higher circulation and the increased amount of advertising, it has been possible to change over to folded A3 format with a glossy cover. The extra costs involved should be amply repaid by the resulting increase in over-the-counter sales, which in turn will help to attract more advertising.

Another change that will be noticeable by its absence, so to speak, will be the reduction in spelling mistookes (oops!) and grammatical lapses due to Tony Williams' invaluable sub-editorial efforts. Tony has even managed to surface occasionally to add his name to some articles of his own.

Once again, the 'typesetting' for the bulk of the magazine has been done on a Centronics 737 printer from Applewriter files, using Ian Trackmans' new printer module. We've had a lot of enquiries about this since the last issue, and are very pleased to say that this is being marketed by Microsense themselves, and should be available from your local dealer (or direct from the author at Blue Chip Software) by the time you read this. Ian is now working on similar modules to interface the Epsom MX80 series, the Paper Tiger, and the Anadex printers with Applewriter.

Our contact with Microsense is bearing fruit - they've promised to keep us in touch with all developments, and are anxious to help us as much as possible. The first example of their help is that a copy of this issue is being sent to every Apple dealer in the U.K.

THE SILENT MAJORITY

The publication in the last issue of the draft constitution with a request for comments would appear to have fallen on stony ground. In the complete absence of any feedback the committee, aided by Ian Trackman who is a lawyer (between writing brilliant software!), are preparing a proposed constitution and agenda for the forthcoming AGM, and details of this will be sent to all members shortly.

PAYMENT IN KIND

Well not exactly in kind, but contributors to Hardcore will receive credits for the Contributed Software Library at the rate of 20 units (sectors) per published page. Provided both circulation and advertising continue to grow, we also hope to start paying a modest page rate to contributors.

The uses to which the Apple can be put are widespread, and (perhaps partly because of this) the ability and the level of knowledge of Apple users is extremely varied. So whatever your own particular interests and capabilities, there are other users out there who can benefit by what you have to say. Some of the articles that come in are in draft or note form, so we also from time to time need people to write these up into a finished article.

One particular area where almost every user can contribute to the journal is in reviewing software. It's in all our interests for both good and bad software to be identified as such.

HELP WANTED

Do we have a photographer member who can help occasionally by producing Black & White prints to our requirements, from negatives and (ideally) also from artwork, video screen, and real-life.

It would also be useful to know of other skills, abilities, and contacts that members can make available in case of need.

INEXPERT ADVICE

T.Tse's article in this issue (Indexed Random Access Method, or IRAM) will probably be incomprehensible to a large number of you. Although we realised this, we felt we should still publish it because we need ONE OF YOU to write a "Beginner's Guide to IRAM". We already have lined up for the next issue an experts review (thanks, Nik Spicer), which again will probably be over the heads of most beginners; so please, somebody redress the balance.

DON'T CALL US...

Some early BASUG literature gave the phone numbers of several committee members. With the continued rapid growth of the group it has become impractical to deal with the ever-increasing number of phone queries. So please WRITE in, to the group's Box number, (enclosing an SAE if you want a reply). You'll get a response as quickly as possible, but we are an entirely voluntary group and you should allow for up to fourteen days response time.

GOING DOWN

Computers are, in the west end of London.

Lion Micro Computers have opened a new and enlarged Computer Book and Magazine section on their ground floor, which in turn means that their demonstration studio on the first floor can now offer a better service to software and hardware customers.

The Sofwarehouse has come down from its Ivory Tower on the third floor of 146 Oxford Street (although Mail Orders will still wing their way from that location). The retail activity is now located in a spacious ground-floor showroom round the corner in Horse Shoe Yard, off Brook Street. Keith Jones, the genial proprietor (a BASUG member, by the way) ably assisted by his assistant Paul, will now convert your unwanted money into useful software, peripherals, etc., six days a week. Keith points out that there's plenty of room to park your wife in nearby Oxford Street while you shop.

BAD NEWS

When the group was started, it was intended that, in addition to the benefits members have already received, a free issue of Apple Orchard would be given. However, with the difficulties of getting bulk supplies at a reasonable price, and the expenses of producing Hardcore (which has grown somewhat from the original news-sheet concept!), we have to place on record the fact that there won't be a free Orchard.

MORE BAD NEWS

The Pascal course, scheduled for May, had to be cancelled at the last minute due to lack of response. In retrospect we probably tried this too early - there were only some forty members at the time it was arranged. If there's a call for it, we intend rescheduling the course for the Autumn. If you would want to take part, or have any comments about when and how it should be held, write in.

GOOD NEWS

Agar Computer Services run a series of two-day courses in BASIC programming, as well as various one-day familiarisation courses. They will give a 10% discount to BASUG members. Details from R.S.Agar-Hutton on 01-328-9232.

WINDFALL

BASUG was formed, only six months ago, because at that time there was no active British user group, and no readily available sources of information for users.

Suddenly, user groups are springing up all over the country (and BASUG is actively helping by publicising these and assisting in forming groups wherever there's a gap).

Now, hard on the heels of Hardcore, comes news of a second specialist Apple publication called 'Windfall'. Unlike Hardcore, which is produced completely on a voluntary basis, this is a commercial proposition, and this will be reflected in the style and presentation, which we understand will be to the standard of "Omni". It is a joint enterprise of Europress and David Chadwick. The latter, who is the Editor, is also an Apple dealer as well as a leading light of the North West Apple Users Group; who, together with other groups in the north-west are said to be giving the new magazine their active help.

BASUG has been approached with a view to Hardcore being absorbed into Windfall, and the group providing information for a User Group Section within Windfall. We felt however that this would neither be to member's wishes nor in the best interest of the group. Hardcore is, to a considerable extent, BASUG; and the reason we get such active and enthusiastic help is presumably because it is a voluntary and non-profit making enterprise.

We wish Windfall well. The proliferation of computer magazines in the states shows that there's plenty of room for everybody; although the standards of some of the magazines over there would suggest that even in that cradle of computing there are problems in filling all that room!

ITT DEVELOPMENTS

Two items of interest concerning graphics. Firstly, ITT have provided us with Pascal disk PAL1, which replaces APPLE1 and corrects Turtlegraphics for the ITT.

Secondly, Ron Davies has provided details of a clever but simple-to-install hardware modification to emulate Apple graphics on an ITT - no more split-down-the-middle Space Invaders. See the next issue for full details.

David Bolton

LOCAL GROUPS

*** We offer space here for Apple user groups, or indeed for any groups catering for Apple users, to publicise themselves.

EAST MIDLANDS

On Wednesday 29th April an inaugural meeting of an Apple user group based on the Leicester area was held. Twelve people attended and a general discussion was followed by the election of a Chairman and Secretary of the group.

It was decided that the group would initially meet on the first Wednesday in the month at the premises of the:-

Leicester Computer Centre
67 Regents Road
Leicester

The meetings will start at 7.30pm and will be open to anyone from the East Midlands Area. The next meetings are detailed below but if anyone would like more information before this date then please contact either:-

Chairman: Mike Preston [illegible]
Secretary: Hazel Bown [illegible]

Next Meetings:

Wednesday June 3rd	Apple Graphics
Wednesday July 1st	Introduction to Machine Language programming

We look forward to seeing any local B.A.S.U.G. members at any of our meetings.

THAMES VALLEY

An inaugural meeting will be held at the

Bisham Abbey Sports Centre
Bisham
Bucks

at 8.00pm on Wednesday 27th May.

This is after the press date of this issue of Hardcore, but up-to-date information can be obtained from:-

Steve Proffitt [illegible]

BRISTOL APPLE USERS AND DABBLERS (B.A.U.D)

This well-established group meet at the premises of:-

Datalink Microcomputer Systems Ltd
10 Waring House
Redcliffe Hill
Bristol BS1 6TB

Tel: [illegible]

Meetings are held on the second Thursday of each month - but check this with Datalink.

BASUG LOCAL CONTACTS

An update to the list published last issue:-

Frank Fletcher	[illegible]
David Row	[illegible]
Leo Crossfield	[illegible]
John Maltby	[illegible]
Len Gould	[illegible]
Geoffrey Clements	[illegible]

Accutrack Disks...

Because data reliability is the important difference in disk construction.

Anatomy of a disk.

Flexible disks are simple information storage devices consisting of a magnetic disk enclosed in a semi-stiff protective jacket. The disk rotates within the jacket while magnetic recording heads on your data or word processing systems "read" or "write" information on the disk's magnetic surface. Since disk operation is simple, it's relatively easy to make one that works. But building in reliability is something else again. It takes specialized technology to build disks that operate flawlessly over an extended period of time.

What counts in disk construction.

Key design objectives for a disk are listed below. How well a disk measures up to these objectives relates directly to the throughput, accuracy and overall costs for your data or word processing system. No disk measures up better than Accutrack.

- The magnetic coating must be precisely formulated and uniformly applied. Imperfections as small as five millionths of an inch cause signal dropouts, data checks and wasted processing time as well as errors.
- The disk surface must be absolutely clean, totally flat and permanently lubricated to prevent excessive head wear with subsequent signal degradation and eventual loss of information. (This is the most critical objective and the one that's most often compromised since poor operating results take a while to show up. It's also the area that most effects the long term reliability of your data.)
- The disk must be free to rotate within its jacket without internal drag to avoid further data checks, excessive processing times and errors.
- The jacket must protect the disk from external contamination and damage. It should also remove microscopic particles of debris from the disk surface before they can damage the disk.

Why you'll never find the best disk bargain in the bargain basement.

While there's little apparent difference between other disks and Accutrak, the performance differences can be substantial. Simply stated, an Accutrack disk is premium priced. But the protection it gives your information; the reliability it provides to your operations; and its substantially longer life make it the best disk buy. After all, the real cost of your operations is constructing and processing the data stored on the disk – not the disk itself. It doesn't make sense to trust that data to anything but the best disk. Accutrack.

BASUG BULK BUYING — 50% off PRICE FOR 10s

£17:50 incl FREE LIBRARY CASE!

(ADD 50p POSTAGE / PACKING)

BRITISH APPLE SYSTEMS USER GROUP

P.O. Box 174, Watford WD2 6NF.

CHAIRMAN'S CORNER

Let me begin by personally thanking all those who helped to make our stand at the Second London Computing Fair such a success. In particular David Bolton and Martin Perry for setting up the stand to make it one of the most professional there.

I should perhaps also welcome all those who joined there, and indeed, those who have heard about us in other ways. The group is now firmly established and is obviously going from strength to strength.

The people I met during my sessions on the stand were a varied group, from the persons looking for a machine to the one who had actually built his own machine from the circuit in the manual!

The varied outlooks, and the different ways in which the machine is being used are beginning to surface. There is a response to questions being asked. My page last month talked about the Autostart Rom problems and, lo and behold!, an article has appeared which looks as if it could solve them. I hope it has managed to reach this issue. We certainly do not have the problem of filling the magazine, only what to leave over until next time. It is not just quantity, it is quality as well.

The beginners may be neglected a little, perhaps because they are afraid to be measured against more sophisticated machine code routines and other highly involved articles. We do have a large number of beginners so if you have written a short ten line program, or discovered how to do something quite simple, someone somewhere is going to thank you for having put it into writing.

Computer users can be compared to car owners. Some just use the machine to get from A to B and let the program take over; e.g. just booting a Visicalc disk means you need know only what you want to do with the huge sheet of paper you are presented with. There are others who want to get into the workings of the machine, dissect it, find out how it works and put it back together again. These may be hardware or software 'mechanics'. But to follow up the analogy with the car people, those who tread the middle way who program with their knowledge of how the machine works and bring together the software skills of others, travel to the best places and enjoy the journey as they go. We have to be thankful there are so many of them about; they are the truly creative ones who help enrich all our lives. The collectors also exist, seizing all software they can, but never really using it. They do not really get the best out of the machine. It is only a toy. We are all overwhelmed with a new acquisition, but the time comes when you realise that you are not really interested in all that software, you start to get selective and delete some of it. The Apple starts to ripen and really yields its fruit when you can use it creatively whether at the problem solving, intellectual level or to actually do something with it. That is not to say that playing games is not good. There are times when we all need to relax and games are the best way to learn and stretch muscles in the brain you didn't know you had, both in writing and playing them. Some of them can be very creative in themselves, almost a new art form.

The main problem with the one lifespan allotted to us, is fitting everything in , if you have to work to pay for such an expensive hobby as well.

This may be the last corner I write for the time being. The AGM will have been held before the next issue is out and it remains to be seen who will be sitting in the Chairmans seat next month. I have enjoyed the initial six months and look forward to serving in some capacity in the coming year as we move on to bigger and better things.

BASUG NEWS

BASUG LEAVES THE NEST

By Tony Williams

On Sunday April 26 BASUG ventured far afield from its burned-in Herts habitat, namely to Covent Garden, at the invitation of the Inter-Varsity Club. This Sunday afternoon "Essential Utilities" bull session was attended by some 60 or so members, many of whom were new faces experiencing their first club meeting after joining at the North London Computer Fair. Broad were the smiles of members living south of the river, grateful at not having to make the long trek across town to St Albans.

The session began in a chaotic fashion, that is to say, far too organized, but after Ian Trackman had gamely kicked off with his "Demo on 1 Monitor For 60 Viewers!" John Sharp managed to break the session up into more manageable units. Club stalwarts demonstrated a variety of utilities useful to all levels of competence, ranging from Richard Teed's presentation of D-patch (how to recover lost machine-code routines on disk - strictly for the advanced D-Appler) down to Vernon Quaintance's exposition of Renumber, Merge and Append. As Vernon said, we might fondly imagine that our members have passed beyond this kind of thing already. Not so - every hands-on demonstration of these fundamental routines finds an avid audience. Perhaps we should coin a BASUG'S Law: "There is always another beginner!"

Frank Kay started out by demonstrating some of the more confusing aspects of the Apple's various cursor control facilities but soon did a shimmy off into his favourite Pascal country. Over in the dark corner Ian Trackman demonstrated two utilities, a mini-assembler and his very own Super Editor, while John Roger by the door showed off a complementary package - Synergistic Software's PLE or Program Line Editor. I am a Super Editor person myself and have recently also discovered PLE, and can hardly think back to those drear and far -off days - was it four, five months ago? - when I was trying to do some serious programming without either of these vital props.

One other facility which quite inadvertantly escalated into a minor attraction was the Spider Software Database on which I was doing on-the-spot registration of new members. I only wanted the facts, sir, but instead had to explain the system instead. Pity I wasn't selling.

Major request from members south of the river and others: could we put them in touch with Apple users living in their vicinity. Answer: Yes we could, and will, either by helping to form local groups or by placing free announcements in "Hard Core". One valuable suggestion came from a new member, Mr Lepley (a former Sorceror owner, new to Apple) that BASUG sub-groups might find it easier to get off the ground by affiliating to existing local hobbyist clubs. His club, for instance, the East London Amateur Computer Club, meets weekly in the Harrow Green Library (Leytonstone - no connection with Harrow!) and would welcome an Apple Users section.

Thanks then to our hosts, the Inter-Varsity Club - some of whose computing members sat in on our session. Bruce Snyder (BASUG member) summed up the afternoon: "This was exactly what we expect a club meeting to be like. No formal presentation piece, but lots of scattered events going on simultaneously, and in the middle knots of people moving about and talking. People who would hesitate to air 'their' problem in front of a large gathering are much more ready to do so in small groups obviously operating on their level."

THE SOFTWARE LIBRARIES

By John Rogers

As B.A.S.U.G continues to gain new members at a steady gallop, likewise the individual sections of the group grow at a similar rate, the software libraries being no exception! This can only be good news to you the user, but it does mean much more work for those running the libraries. It is because of the current situation that the decision to make the libraries virtually a totally postal concern has been taken.

The only two aspects of the libraries that will continue at the main meetings will be, firstly the taking of orders for programs from the Contributed Software Library (CSL) and, secondly the purchase or ordering of disks from the Software Distribution Library (SDL). Offers of programs to the CSL will also continue to be accepted but it would help if (as tends to happen at present) you lend us your disk (or tape) to do what ever is necessary.

In addition, we shall make a postage and packing charge for each order from the CSL received after the end of May. This will take the form of a flat rate of 30p to all members in the U.K., and will be payable at the time of ordering, either in stamps or postal-orders. The cost of SDL discs and tapes already includes postage and packing.

The duplication of programs within the CSL is increasing and highlights the point that if you are going to type in a program specifically for submission to the library, you should contact me before you do.

The next publication of the CSL catalogue will have the heading LENGTH changed to VALUE. The reasons for this are, firstly that a number of programs are in fact a suite of two or more and sometimes are not of the same type (eg. one BASIC and one machine code program). Secondly, that as mentioned in the software libraries article in Hardcore 1, differing types of program carry different values. The value for each program (or suite of programs) is calculated in accordance with the table below.

TYPE	VALUE
BASIC	= 1
Machine Code	= 3
Text Files	= 1
Apple Writer Files *	= 0.5
Binary Graphics File **	= 0.25
Written Text * per page	= 5

(NB. all values are ratios to 1 disk sector, apart from the written text)

* eg. program documentation - a page of written text is taken to be 1 page of A4 paper, hand-written or typed text or drawings.

** File(s) created as a demonstration from a program in the submitted suite.

Finally I would like to take this opportunity to thank all those who have submitted programs to the library and to those who are helping to keep the libraries running. Keep up the good work !

APPLE USER GROUP EUROPE

By Tony Williams

Users of BASUG'S Software Library will have spotted a German disc sent over by the Apple User Group Europe, which, to judge from the information available, should properly be known as Apple User Group Germany since the material concerns German-speaking areas only.

The group is considerably older than BASUG, having been in existence in various incarnations since early 1979 and as of January 1981 it claims 500 members. The Chairman and leading light is W.Dederichs.

Most information about the activities of the group comes in a monthly magazine originally called Apple-Com-Post following in the cherished tradition of execrable apple puns, but now renamed, quite incomprehensibly, P-USER Magazine. The magazine, still edited by Mr Dederichs, presents some Apple chit-chat, publishes lists of Apple hardware available and the Apple dealers who stock it, and prints the names and addresses of User Group members.

It regularly publicises forthcoming meetings of local groups (usually in pubs!) in various parts of Germany, mainly in the north. Some of its listings of programs and utilities, time-saving tips, etc. are original, but most are culled from Call-A.P.P.L.E. and other U.S. sources. The editors are trying to tap the rich vein of grotesque humour latent in Apple users, as evidenced in the call for contributed absurdities to match the following :

```
10 FOR GET 100-200
20 FOR I=0 TO ABOUT 12
30 LET A=1.3 OR 1.4
40 GOTO 100 OR PERHAPS TO 200
```

(taken from BYTE 4/79)

Readers found some difficulty in matching that.

The magazine is naturally replete with anglicisms, (well done! as Henry Root would say) which makes the meat more digestible for any interested non-German speaking BASUG members. For their benefit I have browsed through the articles contributed by Apple User Group Europe members and singled out the following titles which might be of vital importance to you.
I have disregarded U.S. originated materials. Let me know if you are interested and I will arrange to make the material available to you with brief explanations.

1. Assembler program for reading strings containing colons, commas and quotes marks (Apple-Com-Post 1)
2. Analysis of program storage in internal memory (A-C-P 3)
3. Basic Program for Radio Hams to measure distance and direction of other stations (Mr. Boeder.a-c-p 3)
4. Hardware Modification Tips to Avoid Need for DOS 3.3 Basics Disc.
5. DOS 3.3 Disk Map
6. Music! How to calculate semi-tones
7. Printing Hires graphics on NEC printer.
8. Game "Simu Astro" (all P-User Magazine Jan 1981)

In his message to members of January 25, 1981, W. Dederichs mysteriously apologizes for the 'treacherous' demise of Apple-Com-Post which he attributes to pressure of business (on himself!) and lack of contributions from apple users. There are clearly lessons to be learned somewhere here.

THE DISAPPEARING "IF" TRICK

By Ian Trackman

Digital computers (presumably so called because a scientist once poked his finger into a Ram socket to see how it worked) are fundamentally very simple-minded machines, distinguishing only between "Yes" and "No", "True" and "False" or, more specifically, "On" and "Off". The "On" is not a question of whether you have plugged your Apple into the mains socket, but whether a specific location in the processor chip (the 6502 CPU) is or isn't carrying an electrical charge. By the time that a host of clever engineers and computer systems designers have worked on that electrical charge, it has been turned into the Basic "IF" function.

When your Apple finds an "IF", it carries out a series of calculations with the sole purpose of establishing whether or not the relevant part of the chip is electrically charged and therefore, whether the "argument" (that is, the formula, equation or whatever else that comes before the "THEN") is true or false. If the argument is true, Apple celebrates the fact by giving the argument full marks with a score of 1 out of 1. If the argument is not true, that is, if the "IF" test fails, it is awarded 0 out of 1. In the computer's simple On/Off world, there are no part-scores for "nearly true". (Please note that not all Basics adopt the same scoring system - some use 0 for True and -1 for False). Armed with its 0 or 1 result, "IF" then decides whether or not to execute the rest of the statement after the "THEN".

Let's now turn our attention to another Basic function, "NOT". "NOT" is like the compulsive liar in those puzzles of logic about the traveller who meets three strangers at the cross-roads, one who always tells the truth, one who sometimes lies and our friend "NOT", for whom "Yes is always "No", "True" is always "False", "False" is always "True", 0 is always 1 and 1 is always 0. Let's put "NOT" to the test :-

```
IF ( 6 = 2 * 3 ) THEN ....
```

"6 = 2 * 3" is our "argument". We'll keep it enclosed in brackets so that we can recognise it easily. As our argument is true, Basic gives it a score of 1.

```
IF ( 6 = 4 * 7 ) THEN ....
```

is false and gets 0.

```
IF NOT ( 6 = 2 * 3 ) THEN ....
```

Now, "NOT" inverts the situation and we get 0. Finally, :-

```
IF NOT ( 6 = 4 * 7 ) THEN ....
```

produces 1.

We could have put the last two examples in another way, using the "greater than" and "less than" functions, so :-

```
IF ( 6 <> 2 * 3 ) THEN ....
```

and

```
IF ( 6 <> 4 * 7 ) THEN ....
```

but there's method in our madness. (Wait and see !)

At this stage, we can't actually see the 0 or 1 score, since it's internal to the Basic interpreter. What we have established is that, in certain situations, an argument by itself or "NOT" plus an argument will produce a 0 or 1 in the Apple. Let's now try to find it. How about :-

```
PRINT ( 6 = 2 * 3 )
```

and sure enough, Apple replies with a 1. You'll find that it works properly with the other three examples.

Here's a slightly trickier example :-

```
A = 0 : PRINT ( A = 0 )
```

Since the argument "A = 0" is true, the result is 1. Moving on :-

```
A = 4 : PRINT ( A = 4 )
```

and the result is 1 (not 4 !).

If we add in "NOT", so :-

```
A = 0 : PRINT NOT ( A = 0 )
A = 4 : PRINT NOT ( A = 4 )
```

then 1 becomes 0. You can't say :-

```
PRINT <> ( A = 0 )
```

because that produces a syntax error.

What happens if, instead of using an equation as our argument, we use a variable ? The answer is that the True / False test still works. However, we can't demonstrate the "True" condition with a straight-forward PRINT statement, since :-

```
A = 4 : PRINT ( A )
```

simply prints the value of the variable A, which is 4. On the other hand, we can show the "False" case with a "NOT" :-

```
A = 4 : PRINT NOT ( A )
```

which gives us 0. What Basic is doing here is evaluating whether A = 0. If it isn't, we score 1, which is inverted by the "NOT" to 0. If A has some other value, the argument "A = 0" is false and our True / False score is 0. Let's use "IF" to demonstrate the situation where the variable has a value other than 0 :-

```
A = 4 : IF ( A ) THEN PRINT "TRUE"
```

and we get "TRUE".

So here we have the beginnings of a powerful programming tool. If you want to test whether a variable equals 0 or has some other value, you don't have to say :-

IF X <> 0 THEN ....

but simply :-

IF X THEN ....

As you write your ultimate Adventure, you can include statements like :-

IF LAMP AND OIL AND NOT BLIND THEN ....

(LA, OI and BL are, of course, variables). Besides making for shorter code, there is another advantage in this construction - speed.

X = 10 : FOR I = 1 TO 5000 : IF X <> 0 THEN A = X : NEXT

takes approximately 23 seconds, whereas :-

X = 10 : FOR I = 1 TO 5000 : IF X THEN A = X : NEXT

takes only 14 seconds. Using "NOT" instead of "= 0" also produces time savings.

The other major use of this 0 / 1 idea derives from the fact that when you multiply a number by 1, it stays the same and when you multiply it by 0, it becomes 0. Let's justify the title of this article and make the "IF" disappear !

Now you see it :-

IF X = 10 AND Y = 20 THEN Z = 30

and - drum-roll, maestro, please :-

Z = 30 * (X = 10) * (Y = 20)

and now it's gone !

What we are saying in this line of Basic is :-

If X <> 10 and Y <> 20 then :-

Z = 30 * 0 * 0 (i.e. Z = 0)

If X = 10 but Y <> 20 then :-

Z = 30 * 1 * 0 (i.e. Z = 0)

If X <> 10 but Y = 20 then :-

Z = 30 * 0 * 1 (i.e. Z = 0)

If X = 10 and Y = 20 then :-

Z = 30 * 1 * 1 (i.e. Z = 30)

Actually, that wasn't a very good example, since those brackets and multiplication add to Basic's computing time. However, there are two situations where the idea can be of great use. The first is where you want to have the equivalent of several "IF"s followed by another "IF" which applies to all of the cases. One method is to have a number of "IF" statements on different lines, each one followed by a "GOTO" to fall into the next main "IF" statement.

Extending our new method to such statements as

A = 5 * (X = 10) + 10 * (Y = 20) - 5 * (Z = 30)

we can handle the entire series of "IF"s in one go. Notice how we use the plus, minus and multiply operators as the arithmetical equivalents of "IF", "AND", "OR" and "NOT".

The second situation makes the "FN" (function) statement much more powerful. As you know, Applesoft will not handle multi-statement function definitions. It would be useful to be able to say something like :-

```
DEF FN A ( X )
IF B <= 50 THEN FN A = RND ( 1 ) * X
IF B > 50 THEN FN A = RND ( 1 ) * 100
FN END
```

which normally has to be done with a subroutine. But how about :-

DEF FN A (X) = RND (1) * (((B <= 50) * X) + ((B > 50) * 100))

Two final words of caution - using 0 / 1 control logic in a program makes it less portable and harder for others to understand instead of simple "IF .... THEN" constructions. Also, do some bench-mark timing tests if timing is important.

P.S. Now tell your friends that you understand about Boolean algebra !

AMATEUR RADIO

*** This month's copy seems to have gone astray, so here are two items from your good Editor. WARNING! - I write with complete ignorance of Ham Radio.

1. Kevin Hawkins (G8 KIO / A - North London) will be monitoring S20 from 4pm local time Sunday afternoons from 7th June onwards for anyone who would like to set up an Apple Natter group in the London area. Anyone with comments / ideas for times or frequencies please get in touch.

2. Mr Schaefer, Mutterstadter Str. 2, D-6700 Ludwigshafen, proposes an European HAM-meeting, to start the first Sunday of each month at 0900 GMT on 14.260 MHz. His call-sign is DK6UG.

JULY HARDCORE

COPY DATE - 30th JUNE

NEW DEVELOPMENTS IN TECHNOLOGY FOR THE HANDICAPPED CHILD.

By Norah Arnold

Although it has been apparent for some time that 'chip' technology opens up exciting new possibilities for the severely disabled, there has been very little movement towards familiarising teachers of handicapped children with the use of microcomputers. At the Lady Zia Wernher School for physically handicapped children in Luton, however, a series of meetings has been arranged to encourage teachers to implement the new technology by which severely disabled children unable to write in the normal way can record Maths, English and graphics.

Mr. Peter Deakin, Project Co-ordinator for the Spastic Society's Neath Hill Professional Workshop, brought an Apple II with him when he arrived at the school for the first meeting in March. He described how Apple systems have recently been handed over to to six of the Society's schools for spastic children. Incidentally, the money for these Apples came from the recent Blue Peter Appeal fund.

Why had the Apple been chosen? Mr. Deakin looked in detail at what many small systems had to offer, including Nascom, Pet and TRS80 before deciding on the Apple mainly because of the expansion slots.

In order to isolate the problems involved in the use of the Apple by spastics, twelve severely physically handicapped people, all graduates of proved learning ability, have, with Mr. Deakin's help, been trying to implement an all electronic office. Next time you hit the wrong key spare a thought for those among the twelve who are foot operators.

The first obstacle that has to be overcome is the question of interfacing the handicapped child to the computer. A child with limited hand control may be able to manage with a keyboard cover on which they can rest their hand while using a pointer to press the selected key. Often Reset has to be disconnected or linked to separate outside switches. More severely disabled children may have to rely entirely upon some kind of joystick. The Spastic Society have adopted a single plug-in card with internal keyboard which the Apple scans for use by the handicapped child. In this situation speed of operation is a problem. Appropriate software can help by cutting out some commands, or by removing some characters from words. Researchers at King's College, London and at Edinburgh University have co-operated to produce a program called Shiftless. This cuts out all the two-key strokes which would otherwise present a great problem to disabled operators.

On exhibition at the Design Centre, London, are three items which show how great an impact microprocessors will have on the lives of disabled people. The first is a Splink touch sensitive word chart enabling those without speech to communicate via a television screen; the second is a Lightwriter touch sensitive typewriter keyboard, and thirdly an input method designed by Patrick Poon of King's College, London, using only two large touch pads.

In the future it is envisaged that a handicapped child's first toy should be a cheap computer with few controls, enabling the child to use switches to control simple programs. The child would then adapt easily to the use of computers as a classroom resource. Small computers mounted on wheelchairs would be used by the children to enter their individual work. These would then be linked to the classroom's Apple system, and the work dumped out on a printer or stored on a hard disk. It is hoped that intellectually bright handicapped children would be able to use a computerised Stock Control, Pay-Roll and Mailing List by the time they left school.

Mr. Roger Jefcoate, consultant assessor and lecturer on technical aids for disabled people, who also attended the meeting at Lady Zia Wernher School, made an appeal that experts in the field of microcomputers who are interested in helping the disabled should share information as quickly as possible. This may prevent people from re-covering work already done by others.

COMPETITION NEWS

A number of members have asked for clarification of the way in which we'll be calculating the size of the Hex-Decimal / Decimal-Hex routines in our competition.

The number of bytes taken will be the difference between the Applesoft "beginning of program" pointer and "end of program pointer". Don't worry if you don't know how to work this out - just have a go at the routines. If you CALL a routine from the Monitor or the Basic Rom, the bytes used there will not be added to the total.

As a considerable number of new members have joined since the competition was announced, we have decided to extend the closing date until the 30th June to allow them a chance to enter. And to make the competition even more exciting (isn't the glory of winning enough ?), the prize in each category will be two free disks of your choice from the Software Distribution Library. Winners will be announced in the next issue.

Please also let us have your ideas for future competitions. Do you want fun problems or solutions to serious programming difficulties ?

Verbatim®

MINIDISKS

Verbatim, the world's leading producer of minidisks, makes the widest range of minidisks available for word processing and data processing applications. Intent on maintaining this position of leadership, we constantly strive to make our minidisks even better.

Each Verbatim minidisk is equipped with a hub reinforcing ring to aid in registration, reduce errors and give better alignment repeatability.

Try new Verbatim disks for the performance of a lifetime.

SPECIAL OFFER TO B.A.S.U.G. MEMBERS ONLY

10 x MINIDISKETTES **£19·50** INCLUSIVE OF V.A.T. AND POST.
I ENCLOSE A CHEQUE MADE PAYABLE TO B.A.S.U.G. CHEQUE No ..... PLEASE FORWARD TO ME BY RETURN AT THE ENCLOSED ADDRESS.

NAME .....

ADDRESS .....
..... POST CODE

NEW PROGRAMS FOR THE APPLE II®

D/DATABASE

D/DATABASE uses advanced programming techniques and unique data storage and retrieval routines. A special high-speed disk I/O controls the data held on disk, searching and evaluating information at many times the rate achieved by the standard DOS's random access capabilities. Every possible byte on a disk is available for data storage on a DDA formatted disk. D/DATABASE is not operated using limiting numbered indexes. All 'conversation' with the system is in the form of logical statements, similar to BASIC statements.
10 databases per disk maximum - 128 characters maximum record size.
9 character field names - 909 (DOS 3.2) or 1119 (DOS 3.3) useable records per disk.
9 user named fields per record - 27 characters maximum per field within total limit.
16 character index files - D/DATABASE is VERY user-friendly.
£39.95 Basic and machine-code. Requires 48K. Specify DOS 3.2 or 3.3 when ordering.

DDA FILES CONTROLLER (D/DATABASE DISK UTILITIES)

This package controls sorting, copying, restructuring etc of D/DATABASE files. Facilities include:- Copy disk; Copy database; Alpha-sort database (any field etc); Modify format; Re-format and restructure files; Create DOS random access file from database; Move blocks of files + much more.
£39.95 Basic and machine-code. Requires 48K. Specify DOS 3.2 or 3.3 when ordering.

OLDORF'S REVENGE

An exciting hi-res game using over 100 pictures. As you explore the caverns and castles looking for treasure you must battle the one-eyed, two-thumbed Torkie; find the Grezzerlip's sword; visit the Snotgurgle's palace and journey through the domain of the three-nosed Ickyup.
£14.95 on disk only. Requires Applesoft and 48K.

TARTURIAN

Explore 160 rooms (each in hi-res) gathering weapons and treasure that will prepare you for the final battle against the Tarturian. You will encounter deadly Krolls, battle the Minotaur, try and get by COunt Snootweeker, decipher the Yummy Yakky's secret and avoid ghouls.
£19.95 on disk only. Requires Applesoft and 48K.

PROGRAM LINE EDITOR

This program operates in two modes, firstly as a program editor allowing insertion, deletion and modification of any line, and when the changes are complete, hitting RETURN anywhere in the line, adds the now changed line to the program. The second mode is the 'Keyboard Macro' mode. The much neglected 'ESCAPE' key allows you to assign any sequence of characters to any key on the keyboard. Thus, frequently used commands can be reduced to two keystrokes.
£24.95 Machine-code. Available on disk only.

All the above software packages and many more for your Apple system are described in our free catalogue. Please write or telephone for your copy. The prices quoted above include VAT at 15 but please add 50p P+P for orders under £30.00 totally.

SPIDER SOFTWARE

44 LENHAM ROAD, SUTTON, SURREY SM1 4BG
TEL:- 01-661 2365

DOS DIFFERENCES

By John Sharp

Those people who have bought their machine with DOS 3.3, as well as those who have not yet upgraded to 3.3 from 3.2 are not always clear what is the difference between the two, and what they can and cannot do with the DOS they are using.

If you have DOS 3.2, you can only run and initialise DOS 3.2 disks.

If you have DOS 3.3 you can run and initialise DOS 3.2 and DOS 3.3 disks.

If you have the Pascal system, then you have DOS 3.3.

IF YOU HAVE DOS 3.2

The disks you are using have 13 sectors on 35 tracks. Three of these are normally used to store the DOS so that you can boot from the disk, and one to store the catalog or directory information. This leaves 31 tracks to store your programs on, giving you 31*13 = 403 sectors = 101K of memory. (Each sector holds 256 bytes.)

If you wish to update to DOS 3.3 then you must buy a DOS 3.3 UPDATE KIT which consists of two chips, the P5A and P6A proms which replace the P5 and P6 proms on the Disk controller card. There is also a copy of the DOS 3.3 manual, and two Disks, a 3.3 MASTER and a BASICS disk. It is possible (but not necessarily easy) to get hold of the two proms seperately, but you would still need a 3.3 MASTER disk in order to boot and initialize 3.3 disks.

IF YOU HAVE THE LANGUAGE CARD OR PASCAL SYSTEM

When you bought the PASCAL system, you had to replace the P5 and P6 proms with the P5A and P6A proms, if you had a DOS 3.2 system. You would also have had a BASICS disk the function of which is to allow you to boot a DOS 3.2, i.e. 13 sector disk. In order to run DOS 3.3, all you need now is a copy of the 3.3 master disk.

If you have bought a language card or Ramcard separately and are still running DOS 3.2 and have not replaced the P5 and P6 proms with the P5A and P6A versions, you may have trouble running certain programs and languages. You will only get the full benefits of the card by having DOS 3.3.

IF YOU HAVE DOS 3.3

The disks you are using have 16 sectors on 35 tracks, but with the directory and DOS on the disk there are only 31 available, giving 31*16=496 sectors =124K of storage.

If you wish to run a DOS 3.2 disk, you can either run the BOOT13 program on the MASTER disk, or run the BASICS disk which is essentially the same. There seems to be a problem with a few programs, if you try to run them from a DOS 3.2 disk through the BOOT13 program. If you know of any then write in; someone else may need to know, or they may know why.

If you have got a DOS 3.2 disk booted up this way, and you initialise a disk it is now a DOS 3.2 slave disk. (If you think you have not got one then you can use the INTRODUCTORY DISK. These are all DOS 3.2 at the moment precisely because everyone can run a DOS 3.2 disk as it stands. The reverse is not true, you need the P6A prom on the controller cardto run a DOS 3.3 disk. You need the P5A to boot up on DOS 3.3, but not to run it.)

If you want to move a program from a 13 sector disk to a 16 sector disk, then there is a MUFFIN program on the 3.3 MASTER which enables you to do this. It requires DOS 3.3 to be active. To reverse the procedure, use the PUFFIN program on DISK 3 of the SOFTWARE LIBRARY (also available as a single program in the CONTRIBUTED SOFTWARE LIBRARY). This requires DOS 3.2 active. There are also DMUFFIN and NIFFUM which serve the same function. These are also in the CONTRIBUTED SOFTWARE LIBRARY.

*******WARNING*******

There are a number of utilities in the SOFTWARE LIBRARY, which came in as DOS 3.2, which only work with DOS 3.2 active. If you get a disk which is DOS 3.2 (this means all available at present, i.e. up to number 16) from the SOFTWARE LIBRARY then you could irreversibly corrupt a DOS 3.3 disk if you are using a DOS 3.2 Utility which writes to the disk. This does not apply to normal programs; they can be MUFFINed over and used with no trouble.

AN INDEXED RANDOM ACCESS METHOD FOR THE APPLE II

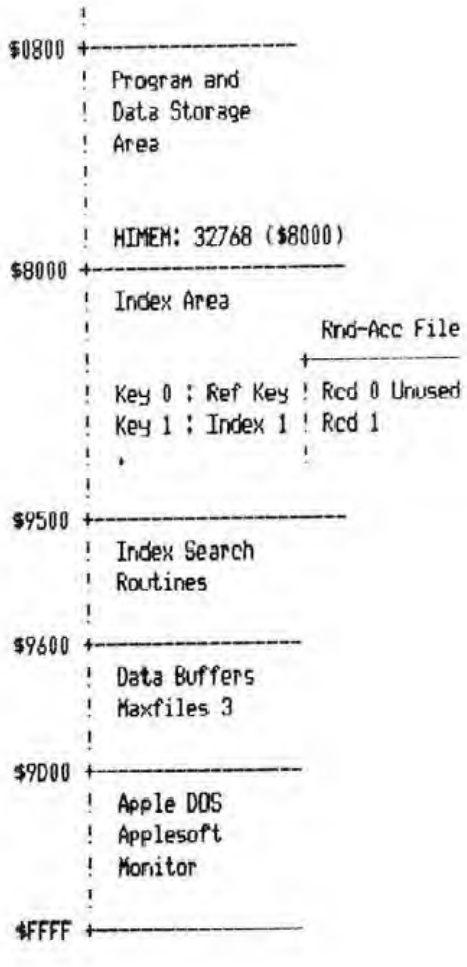
"99% of business programming involves simply storing and retrieving information"
(Nik Spicer - Hard Core Vol 1 No 1)

How true! Data storage on any computer system is generally not a problem at all. The slow and tedious work normally involves the search and retrieval of the damn information. Anybody who has written a stock control or name and address system will know exactly what I mean.

The index search routines presented here provide a simple and fast indexing method for looking up data records stored in a Random-access text file. For most applications, an index search and data retrieval time of less than five seconds will be achieved.

The machine code routines written for a 48K Apple II system interface with Applesoft programs via the usual CALLs. The codes, and indeed the index storage area, are situated in the high part of memory (Fig. 1), and should be protected from Basic by resetting HIMEM: to an appropriate value.

Fig. 1. System Memory Map



Notice also that the index keys are stored as a Binary file on disk - the obvious advantage here being the fast BLOAD and BSAVE of the index data. Within the index storage area itself, index key 'ZERO' is used to store the reference key for the search, and is used internally by the machine code routines quite often. The second index key, referred to as index key 'ONE' indexes the data stored at Relative Record Number '1' of the Random-access file. Program 1 lists a typical HELLO boot program to partition the Apple II Memory area before linking to a main application program.

The routines written, though not transportable to other parts of memory, are flexible enough to allow easy user modification of the index area size, provided certain strict rules are observed. For convenience, several variables can be defined at this stage.

```
INDEX = 8 * 4096
        Start Of Index Area
        Reference Key Stored Here
        (Also HIMEM:)

CODES = 9 * 4096 + 5 * 256
        Start Of M/C Code Routines

MAXNO = 330
        Max No Of Keys In Index

RCDNO = PEEK(CODES + 31) * 256
         + PEEK(CODES + 30)
        Record Number Being Indexed
```

The machine code routines have five main entry points for easy interfacing with Applesoft programs. These are detailed and commented upon below.

1. CLRALL - (CALL CODES) - $9500

This clears the whole of the index area to nulls. To be used with care!!

2. FREKEY - (CALL CODES+3) - $9503

This call returns the record number RCDNO of the first occurrence of a free index key (null key). If RCDNO > MAXNO then the index area is full.

3. FINKEY - (CALL CODES+6) - $9506

This routine tries to match the reference key against one stored in the index area. If RCDNO > MAXNO then no key match has been made, and this is usually an error condition. If RCDNO <= MAXNO then a match HAS been made at RCDNO. On an Add option, this implies a DUPLICATE key, while on an Amend option, the data stored at RCDNO can be read in and displayed in seconds.

4. REPKEY - (CALL CODES+9) - $9509

This routine moves the reference key into the index area pointed to by RCDNO.

5. DELKEY - (CALL CODES+12) - $950C

This last routine deletes the reference key specified from the index area at key RCDNO. The index key is reset to nulls.
It is left to the user to ascertain that the RCDNO parameter passed to the machine code routines (in 4 & 5 above) is valid for these operations. Failure to comply may scramble the index area.

The size of the index storage area can be modified at will to suit most applications. For this the user will need to provide the following parameters to the codes. This data portion occupies only five bytes of memory, starting at address $9510.

```
KEYST   $9510-11 : $00 $80
        Start Of Index Key Storage Area
        In Reverse Hex 6502 Format

KEYLEN  $9512    : $10
        Length Of Each Index Key

KEYMX   $9513-14 : $4B $01
        Max No Of Keys In Index PLUS 1
        In Reverse Hex 6502 Format
```

As a general rule,

```
KEYMX   = CODES($9500) - KEYST
          ------------------
                KEYLEN

HIMEM:  = KEYST
```

Finally, the source listing of the machine code routines used in the index search is included here for the advanced programmer who intends to improve on them by adding sorting and multiple key search routines. The listing of an Applesoft program to demonstrate the index search facilities also follows.

For the less experienced programmer, a demonstration diskette containing all the necessary software is available from the Contributed Software Library. (Please contact John Rogers).

```
1 : REM  APPLE II 48K
2 : REM  INDEX SEARCH DEMO
3 : REM  WRITTEN BY  T TSE
4 :
5 : REM  (C) COPYRIGHT 28-JAN-1981
6 : REM  ALL RIGHTS RESERVED
7 :
10  TEXT : HOME : SPEED= 255
20  PRINT  CHR$ (4)"MAXFILES 3"
30  HIMEM: 8 * 4096
40  PRINT  CHR$ (4)"BLOAD INXCODES,A$9500"
50  PRINT  CHR$ (4)"BLOAD INDEXKEYS,A$8000"
60  PRINT  CHR$ (4)"RUN DEMOPROG"
```

```
1 : REM  APPLE II 48K
2 : REM  INDEX SEARCH DEMO
3 : REM  WRITTEN BY  T TSE
4 :
5 : REM  (C) COPYRIGHT 28-JAN-1981
6 : REM  ALL RIGHTS RESERVED
7 :
10 INDEX = 8 * 4096: REM  START OF INDEX
12 CODES = 9 * 4096 + 5 * 256: REM  START OF M/C CODES
14 MAXNO = 330: REM  MAX NO OF KEYS
16 KEYLN = 16: REM  INDEX KEY LENGTH
19 :
20  TEXT : HOME : SPEED= 255
22  VTAB 1: PRINT "INDEX SEARCH DEMO PROGRAM"
24  VTAB 2: PRINT "WRITTEN BY  T TSE"
26  VTAB 3: PRINT "================"
28  VTAB 5: PRINT "VERSION 1.0"
30  VTAB 6: PRINT "28-JAN-1981"
32  VTAB 10: PRINT "PLEASE SELECT AN OPTION"
34  VTAB 12: PRINT "1. ADD NEW DATA"
36  VTAB 13: PRINT "2. AMEND DATA"
38  VTAB 14: PRINT "3. DELETE DATA"
40  VTAB 15: PRINT "4. INITIALISE DATA"
42  VTAB 16: PRINT "5. TERMINATE SYSTEM"
49 :
50  VTAB 19: PRINT "WHICH (1-5)"
55  VTAB 19: HTAB 12: INPUT A$
60  IF A$ = "" OR  LEN (A$) > 1 THEN 55
65  IF A$ < "1" OR A$ > "5" THEN 55
70  ON  VAL (A$) GOSUB 1000,2000,3000,4000,5000
75  GOTO 20
97 :
98 : REM  INTERFACE ROUTINES
99 :
100  FOR I = 0 TO KEYLN - 1: POKE INDEX + I,0: NEXT I:
RETURN : REM  CLEAR REF KEY
110 RCDNO =  PEEK (CODES + 31) * 256 +  PEEK (CODES + 3
0): RETURN : REM  RECORD NUMBER
120  GOSUB 100: FOR I = 0 TO  LEN (I$) - 1: POKE INDEX
+ I, ASC ( MID$ (I$,I + 1,1)): NEXT I: RETURN : REM  ST
ORE A REF KEY
```

```
197 :
198 : REM  INTERFACE CALLS
199 :
200  CALL CODES: RETURN : REM  CLEAR ALL INDEX KEYS (CA
RE!!)
210  GOSUB 100: CALL CODES + 3: GOSUB 110: RETURN : REM
  FINDS RCD FOR A FREE KEY
220  GOSUB 120: CALL CODES + 6: GOSUB 110: RETURN : REM
  FINDS RCD FOR REF KEY
230  CALL CODES + 9: RETURN : REM  SAVE REF KEY INTO IN
DEX
240  CALL CODES + 12: RETURN : REM  DELETE INDEX KEY
297 :
298 : REM  SCREEN DISPLAY ROUTINES
299 :
300  TEXT : HOME : VTAB 1: INVERSE : PRINT A$: NORMAL
305  VTAB 4: PRINT "    KEY   ................"
310  VTAB 7: PRINT "   DESC   ..................."
315  VTAB 9: PRINT " DATA 1   ..........."
320  VTAB 10: PRINT " DATA 2   ..........."
325  VTAB 11: PRINT " DATA 3   ..........."
330  VTAB 13: PRINT "  OTHER   ......."
335  VTAB 17: PRINT "CONFIRM   .": RETURN
349 :
350  TEXT : HOME : VTAB 1: INVERSE : PRINT A$: NORMAL
355  VTAB 4: PRINT "    KEY   "A0$
360  VTAB 7: PRINT "   DESC   "A1$
365  VTAB 9: PRINT " DATA 1   "A2$
370  VTAB 10: PRINT " DATA 2   "A3$
375  VTAB 11: PRINT " DATA 3   "A4$
380  VTAB 13: PRINT "  OTHER   "A5$
385  VTAB 17: PRINT "CONFIRM   .": RETURN
397 :
398 : REM  INPUT A KEY
399 :
400  VTAB 4: HTAB 11: INPUT "";A0$
405  IF A0$ = "" THEN  POP : REM  TO MENU
410  IF  LEN (A0$) > 16 THEN A0$ =  LEFT$ (A0$,16)
415  VTAB 4: HTAB 11: PRINT A0$;: CALL  - 868
420  RETURN
497 :
498 : REM  INPUT DATA
499 :
500  VTAB 7: HTAB 11: INPUT "";AA$: IF AA$ < > "" THEN
 A1$ = AA$
505  IF  LEN (A1$) > 20 THEN A1$ =  LEFT$ (A1$,20)
507  VTAB 7: HTAB 11: PRINT A1$;: CALL  - 868
510  VTAB 9: HTAB 11: INPUT "";AA$: IF AA$ < > "" THEN
 A2$ = AA$
515  IF  LEN (A2$) > 10 THEN A2$ =  LEFT$ (A2$,10)
517  VTAB 9: HTAB 11: PRINT A2$;: CALL  - 868
520  VTAB 10: HTAB 11: INPUT "";AA$: IF AA$ < > "" THE
N A3$ = AA$
525  IF  LEN (A3$) > 10 THEN A3$ =  LEFT$ (A3$,10)
527  VTAB 10: HTAB 11: PRINT A3$;: CALL  - 868
530  VTAB 11: HTAB 11: INPUT "";AA$: IF AA$ < > "" THE
N A4$ = AA$
535  IF  LEN (A4$) > 10 THEN A4$ =  LEFT$ (A4$,10)
537  VTAB 11: HTAB 11: PRINT A4$;: CALL  - 868
540  VTAB 13: HTAB 11: INPUT "";AA$: IF AA$ < > "" THE
N A5$ = AA$
545  IF  LEN (A5$) > 7 THEN A5$ =  LEFT$ (A5$,7)
547  VTAB 13: HTAB 11: PRINT A5$;: CALL  - 868
550  RETURN
```

APPLE ORCHARD

To subscribe direct write to :-

APPLE ORCHARD SUBSCRIPTIONS
PO BOX 1493
BEAVERTON
OR 97075
USA

The subscription is $10 per year, for 4 issues, published quarterly, with overseas postage of $10 giving a total of $20. The Bankers Draft or Money order in $USA should be made ot to "THE INTERNATIONAL APPLE CORE ". You can get a Bankers Draft from your bank, but it will cost you about £ 3; you can get an International Money Order from The Post Office, again at a colossal fee. The cheapest way is to find a branch of BARCLAYS BANK INTERNATIONAL, which will make you out an International Money Order over the counter for about 60p.

HELP WANTED

MAGAZINE ... SOFTWARE ...
MEETINGS ... LIBRARY ...
EXHIBITIONS ... HELP & ADVICE
... SPECIAL OFFERS ... ETC
... ETC ...

Q. Where does all this stuff come from ?

A. From You - or some of you !

BASUG is a voluntary group, and all these services are provided BY, as well as FOR, the members themselves.

So if you can help in any way, please get in touch.

```
597 :
598 : REM  CONFIRM 'Y' OR 'N'
599 :
600  GOSUB 350: REM  RE-DISPLAY
605  VTAB 17: HTAB 11: INPUT "";A6$
610  IF A6$ = "" THEN 605
612  IF  LEN (A6$) > 1 THEN 600
615  IF A6$ = "Y" OR A6$ = "N" THEN  RETURN
620  GOTO 605
697 :
698 : REM  WRITE A RECORD
699 :
700  PRINT  CHR$ (4): GOSUB 230: REM  SAVE KEY INTO IND
EX
705  PRINT  CHR$ (4)"OPEN DATAFILE,L64"
710  PRINT  CHR$ (4)"WRITE DATAFILE,R"RCDNO
715  PRINT A1$: PRINT A2$: PRINT A3$: PRINT A4$: PRINT
A5$
720  PRINT  CHR$ (4)"CLOSE":UF = 1: RETURN : REM  UPDAT
ED FLAG
797 :
798 : REM  READ A RECORD
799 :
800  PRINT  CHR$ (4)
805  PRINT  CHR$ (4)"OPEN DATAFILE,L64"
810  PRINT  CHR$ (4)"READ DATAFILE,R"RCDNO
815  INPUT A1$,A2$,A3$,A4$,A5$
820  PRINT  CHR$ (4)"CLOSE": RETURN
997 :
998 : REM  ADD NEW DATA
999 :
1000 A$ = "ADD NEW DATA": GOSUB 300
1005  GOSUB 210: REM  FIND A FREE SPACE
1010  IF RCDNO > (MAXNO) THEN 1500: REM  NO SPACE
1015  GOSUB 400: REM  INPUT KEY
1020 I$ = A0$: GOSUB 220: REM  INDEX CHECK
1025  IF RCDNO < = (MAXNO) THEN 1600: REM  DUPLICATE K
EY
1030 A1$ = "":A2$ = "":A3$ = "":A4$ = "":A5$ = ""
1035  GOSUB 500: REM  INPUT REST OF DATA
1040  GOSUB 600: REM  GET CONFIRMATION
1045  IF A6$ = "N" THEN 1015
1047  GOSUB 210: GOSUB 120
1050  GOSUB 700: REM  WRITE RECORD
1055  GOTO 1000: REM  REDO
1497 :
1498 : REM  NO SPACE IN INDEX
1499 :
1500 A$ = "INDEX SPACE EXHAUSTED - HIT RETURN KEY"
1505  VTAB 23: HTAB 1: INVERSE : PRINT A$;: NORMAL
1510  GET A$: IF  ASC (A$) < > 13 THEN 1510
1515  POP : RETURN
1597 :
1598 : REM  DUPLICATE KEY ENTRY
1599 :
1600 A$ = "DUPLICATE KEY FOUND - HIT RETURN KEY"
1605  VTAB 23: HTAB 1: INVERSE : PRINT A$;: NORMAL
1610  GET A$: IF  ASC (A$) < > 13 THEN 1610
1615  GOTO 1000
1997 :
1998 : REM  AMEND DATA
1999 :
2000 A$ = "AMEND DATA": GOSUB 300
2005  GOSUB 400: REM  INPUT KEY
2010 I$ = A0$: GOSUB 220: REM  INDEX CHECK
2015  IF RCDNO > (MAXNO) THEN 2500: REM  KEY NOT FOUND
2020  GOSUB 800: REM  READ RECORD
2025  GOSUB 350: REM  DISPLAY RECORD
2030  GOSUB 600: REM  CONFIRM OK
2035  IF A6$ = "N" THEN 2000
2040  GOSUB 500: REM  AMEND DATA
2045  GOSUB 600: REM  CONFIRM AGAIN
2050  IF A6$ = "N" THEN 2040
2055  GOSUB 700: REM  RE-WRITE RECORD
2060  GOTO 2000: REM  REDO
2497 :
2498 : REM  KEY NOT FOUND
2499 :
2500 A$ = "KEY NOT FOUND - HIT RETURN KEY"
2505  VTAB 23: HTAB 1: INVERSE : PRINT A$;: NORMAL
2510  GET A$: IF  ASC (A$) < > 13 THEN 2510
2515  GOTO 2000
2997 :
2998 : REM  DELETE DATA
2999 :
3000 A$ = "DELETE DATA": GOSUB 300
3005  GOSUB 400: REM  INPUT KEY
3010 I$ = A0$: GOSUB 220: REM  INDEX CHECK
3015  IF RCDNO > (MAXNO) THEN 3500: REM  KEY NOT FOUND
3020  GOSUB 800: REM  READ RECORD
3025  GOSUB 350: REM  DISPLAY RECORD
3030  GOSUB 600: REM  CONFIRM OK
3035  IF A6$ = "N" THEN 3000
3040  GOSUB 240: REM  DELETE INDEX KEY
3045  GOTO 3000: REM  REDO
3497 :
3498 : REM  KEY NOT FOUND AGAIN
3499 :
3500 A$ = "KEY NOT FOUND - HIT RETURN KEY"
3505  VTAB 23: HTAB 1: INVERSE : PRINT A$;: NORMAL
3510  GET A$: IF  ASC (A$) < > 13 THEN 3510
3515  GOTO 3000
3995 :
3996 : REM  CLEAR INDEX AREA
3997 : REM  BSAVE EMPTY INDEX
3998 : REM  INITIALISE DATAFILE
3999 :
4000  PRINT  CHR$ (4): GOSUB 200
4005  PRINT  CHR$ (4)"BSAVE INDEXKEYS,A$8000,L$1500"
4010  PRINT  CHR$ (4)"OPEN DATAFILE,L64"
4015  FOR I = 1 TO MAXNO
4017  VTAB 21: PRINT "DATA RCD : "I
4020  PRINT  CHR$ (4)"WRITE DATAFILE,R"I
4025  FOR J = 1 TO 5: PRINT "": NEXT J: PRINT  CHR$ (4)
: NEXT I
4030  PRINT  CHR$ (4)"CLOSE": RETURN
4996 :
4997 : REM  TERMINATE SYSTEM
4998 : REM  BSAVE INDEX TO DISK
4999 :
5000  PRINT  CHR$ (4): IF  NOT UF THEN 5010
5005  PRINT  CHR$ (4)"BSAVE INDEXKEYS,A$8000,L$1500"
5010  END
```

```
ID=00

      TSE COMPUTERS APPLE IC ASSEMBLER PAGE 01


0010:                    ID=01
0020:
0030:                    APPLE II       (C) COPYRIGHT
0040:                    INDEX SEARCH   28-JAN-1981
0050:                    WRITTEN BY     T TSE
0060:
0070: 9500                       ORG    $9500
0080:
0090: 9500 4C 8E 95  ENT1    JMP    CLRALL !TO CLEAR ALL KEYS
0100: 9503 4C 9D 95  ENT2    JMP    FREKEY !TO FIND A FREE KEYSPACE
0110: 9506 4C A0 95  ENT3    JMP    FINKEY !TO SEARCH FOR SPECIFIED KEY
0120: 9509 4C AC 95  ENT4    JMP    REPKEY !TO REPLACE SPECIFIED KEY
0130: 950C 4C AF 95  ENT5    JMP    DELKEY !TO DELETE SPECIFIED KEY
0140: 950F 00                =      $00
0150:
0160: 9510           ZERO    *      $0000  !NUMERIC ZERO
0170: 9510           INDEX   *      $00D0  !PAGE ZERO INDEX
0180: 9510           KEYSTR  *      $8000  !START OF KEY STORAGE
0190:
0200: 9510 00        KEYSTL  =      KEYSTR !DEFINE START OF KEY STORAGE
0210: 9511 80        KEYSTH  =      KEYSTR /
0220: 9512 10        KEYLEN  =      $10    !DEFINE KEY LENGTH IN HEX
0230: 9513 4B        KEYMXL  =      $4B    !DEFINE MAX+1 NO OF KEYS IN INDEX
0240: 9514 01        KEYMXH  =      $01    !L=LOWBYTE H=HIGHBYTE
0250: 9515 00                =      $00
0260: 9516 00                =      $00    !THIS IS RECORD NUMBER COUNTER
0270: 9517 00                =      $00    !INITIALLY SET TO ZERO
0280:
0290: 9518 00        STPARL  =      $00    !KEY START PARAMETER (LOW)
0300: 9519 00        STPARH  =      $00    !KEY START PARAMETER (HIGH)
0310: 951A 00        LENPAR  =      $00    !KEY LENGTH PARAMETER
0320: 951B 00        COUNTL  =      $00    !MAX COUNT PARAMETER (LOW)
0330: 951C 00        COUNTH  =      $00    !MAX COUNT PARAMETER (HIGH)
0340: 951D 00                =      $00
0350: 951E 00        RTRCDL  =      $00    !RETURN RECORD NUMBER (LOW)
0360: 951F 00        RTRCDH  =      $00    !RETURN RECORD NUMBER (HIGH)
0370:
0380: 9520 A9 00     CLRKEY  LDAIM  KEYSTR !CLEAR REFERENCE KEY
0390: 9522 A2 80             LDXIM  KEYSTR /
0400: 9524 85 D0             STAZ   INDEX  !SAVE IN PAGE ZERO FOR INDEX
0410: 9526 86 D1             STXZ   INDEX  +01
0420: 9528 A9 00     CLRONE  LDAIM  ZERO   !CLEAR A KEYSPACE
0430: 952A AC 1A 95          LDY    LENPAR !GET PARAM FOR INDEX
0440: 952D 88                DEY           !MUST DO THIS
0450: 952E 91 D0     CLRLOP  STAIY  INDEX  !CLEAR KEY BYTE
0460: 9530 88                DEY           !READY FOR NEXT BYTE
0470: 9531 10 FB             BPL    CLRLOP !LOOP TILL FINISHED
0480: 9533 60                RTS
0490:
0500: 9534 A0 07     TRANSF  LDYIM  $07    !TRANSFER KEY PARAMETERS
0510: 9536 B9 10 95  TRNLOP  LDAAY  KEYSTL !LOAD KEY DATA
0520: 9539 99 18 95          STAAY  STPARL !SAVE KEY PARAMETERS
0530: 953C 88                DEY           !READY FOR NEXT BYTE
0540: 953D 10 F7             BPL    TRNLOP !LOOP TILL FINISHED
0550: 953F 60                RTS
```

```
0560:

      TSE COMPUTERS APPLE II ASSEMBLER PAGE 02


0570: 9540 18        PNTUPD CLC          !POINTERS UPDATE ROUTINE
0580: 9541 AD 1A 95         LDA   LENPAR !GET LENGTH BYTE
0590: 9544 6D 18 95         ADC   STPARL !ADD TO ABS KEY ADDRESS
0600: 9547 8D 18 95         STA   STPARL !SAVE IT BACK
0610: 954A 85 D0            STAZ  INDEX  !SAVE FOR INDEX
0620: 954C 90 03            BCC   ADDCHK !CHECK FOR ADDRESS PAGE BOUNDS
0630: 954E EE 19 95         INC   STPARH !YES- PAGE BOUNDARY
0640: 9551 AD 19 95  ADDCHK LDA   STPARH !NEED THE HIGH-BYTE ADDRESS
0650: 9554 85 D1            STAZ  INDEX  +01 !FOR INDEX
0660: 9556 EE 1E 95         INC   RTRCDL !INCREMENT RECORD COUNTER
0670: 9559 D0 03            BNE   RCDCHK !CHECK FOR RECORD NUMB BOUNDS
0680: 955B EE 1F 95         INC   RTRCDH !YES- NUMBER BOUNDARY
0690: 955E AD 1E 95  RCDCHK LDA   RTRCDL !CHECK FOR MAX RECORDS
0700: 9561 CD 1B 95         CMP   COUNTL !COMPARE LOW TO SPECS
0710: 9564 D0 1A            BNE   RETURN !STILL OK SO FAR
0720: 9566 AD 1F 95         LDA   RTRCDH !NOW FOR HIGH BYTE
0730: 9569 CD 1C 95         CMP   COUNTH !COMPARE HIGH TO SPECS
0740: 956C D0 12            BNE   RETURN !STILL OK OK SO FAR
0750: 956E F0 0E            BEQ   POPRTN !ERROR- MAX RECORD NUMBER HERE
0760:
0770: 9570 AC 1A 95  COMPAR LDY   LENPAR !KEY COMPARISON
0780: 9573 88               DEY          !MUST
0790: 9574 B1 D0     COMPIT LDAIY INDEX  !LOAD A KEY BYTE
0800: 9576 D9 00 80         CMPAY KEYSTR !CHECK AGAINST REF KEY
0810: 9579 D0 05            BNE   RETURN !UNLUCKY- RETURN FOR NEXT
0820: 957B 88               DEY          !MUST DO ALL BYTES
0830: 957C 10 F6            BPL   COMPIT !LOOP TILL DONE
0840: 957E 68        POPRTN PLA          !ALL BYTES CHECK OK
0850: 957F 68               PLA          !POP SUB ADDRESS TO MAIN CALLING PROG
0860: 9580 60        RETURN RTS          !STRAIGHTFORWARD RETURN
0870:
0880: 9581 AC 1A 95  REPLCE LDY   LENPAR !REPLACE REF KEY INTO INDEX
0890: 9584 88               DEY          !AGAIN MUST
0900: 9585 B9 00 80  REPLOP LDAAY KEYSTR !GET A REF KEY BYTE
0910: 9588 91 D0            STAIY INDEX  !SAVE INTO INDEX AREA
0920: 958A 88               DEY          !REPEAT FOR ALL BYTES
0930: 958B 10 F8            BPL   REPLOP !LOOP TILL FINISHED
0940: 958D 60               RTS
0950:
0960:
0970: 958E 20 20 95  CLRALL JSR   CLRKEY !CLEAR REFERENCE KEY
0980: 9591 20 34 95         JSR   TRANSF !TRANSFER PARAM DETAILS
0990: 9594 20 40 95  CLRLUP JSR   PNTUPD !UPDATE RECORD POINTERS
1000: 9597 20 28 95         JSR   CLRONE !CLEAR THIS RECORD
1010: 959A 4C 94 95         JMP   CLRLUP !LOOP TILL ALL DONE
1020:
1030: 959D 20 20 95  FREKEY JSR   CLRKEY !CLEAR REF KEY TO SEARCH
1040: 95A0 20 34 95  FINKEY JSR   TRANSF !TRANSFER PARAM DETAILS
1050: 95A3 20 40 95  COMLOP JSR   PNTUPD !UPDATE RECORD POINTERS
1060: 95A6 20 70 95         JSR   COMPAR !COMPARE REF KEY TO INDEX
1070: 95A9 4C A3 95         JMP   COMLOP !LOOP TILL ALL DONE
1080:
1090: 95AC 4C 81 95  REPKEY JMP   REPLCE !REPLACE REF KEY INTO INDEX
1100:
1110: 95AF 4C 28 95  DELKEY JMP   CLRONE !CLEAR INDEX KEY
1120:
```

BEGINNERS' PAGES

By John Sharp

PEEKING AND POKING

To a beginner Basic is quite a mindfull to deal with. It does have some connection with the English language, in that commands such as PRINT mean something related to their everyday use. The two commands PEEK and POKE have an aura of mystery about them, however. They are in fact used precisely as their name suggests. PEEK lets you look at something, and POKE lets you push something into a certain location. This word 'location' is the key to what is happening. The computer memory (be it RAM or ROM) is a set of locations like pigeon holes which store numbers, which are processed by the microprocessor (the brain at the heart of any micro) to carry out the instructions you tell it by means of your program. The microprocessor (in the Apple the 6502 microprocessor chip) runs along these memory locations executing the instructions it sees in each of the pidgeon holes. Some of the locations have specific tasks as switches to set various conditions, for example to put you on the graphics instead of the text screens. Others are actual parts of a program.

When you see a POKE, there are always two numbers separated by a comma. POKE 50,255 means put the number 255 into the 50th location in the memory.

If you see POKES in a program it is important to recognize the context in which they are being used, especially if you want to convert programs written for other machines. In general there are two ways you might see one or more Pokes in a program line. The first uses Basic to write a machine code routine into memory. This is usually a list of Pokes such as:-
POKE 768,1: POKE 769,32: POKE 770,221: POKE 771,221: POKE 772,206: POKE 773,0: POKE 774,3: POKE 775,248: POKE 776,96

and often is CALLED by the program elsewhere. The CALL needs to be followed by a number which tells the computer where to start in the memory, and is very much like going to a subroutine, except that it is a machine code subroutine. The number will often be one of the early numbers in the set of Pokes. In the case above a CALL769 will evoke a response; if you want to know what happens try it and see. If you see this type of example in a PET program for instance you would need to know some machine code to understand what it is doing in order to convert it. It may be possible to use it directly, but normally it will have to be rewritten. A shape table may also be Poked in as a set of numbers in the same way.

The other way you might see a Poke is in isolation. This is more often than not setting a switch. In a PET program you would have to know exactly what it did before you could find the equivalent Poke or command to use on the Apple. The location 50 can be used on th Apple as a switch to set whether a character that is printed is shown as NORMAL, INVERSE, or FLASHing. You can in fact use these words as commands in Applesoft, but not in Integer Basic, so if you wish to use these effects in an Integer Basic program, you have to use Pokes to flick the switches. If you put the number 63 into location 50 with a Poke then the characters printed next will be printed INVERSE. If you put 127 there they will be printed FLASHING. To get back to NORMAL it is necessary to POKE 255 into location 50. The following short program demonstrates this:-

```
10 POKE 50,255 : REM SETS TO NORMAL
20 GOSUB 100
30 POKE 50,63:REM INVERSE
40 GOSUB 100
50 POKE 50,127:REM FLASHING
60 GOSUB 100
70 POKE 50,255:REM SET BACK TO NORMAL
80 END
100 PRINT "THIS IS A STRING "
110 RETURN
```

You might try putting other numbers (up to 255) in this location, to see what happens. There may be a load of rubbish or there may be nothing printed out. Only the three numbers 63,127 and 255 are predictable. Numbers will not always flash or come out inverse.

The POKE 33,33 and POKE 33,28 described in the previous two 'HARDCORE' Beginners pages are further examples of using POKES to set a switch. In this case they set the text width across the screen.

PEEK into this page next month for more POKES and to learn about PEEKS.

PROBLEMS WITH OVERWRITING THE HIGH-RES PAGES

Have you had the end of your program mysteriously disappear when you tried to run it? If it was a very long program, it is quite likely that you cleared or used Hi-Res page one. If the program ran into that area of memory then all code you put in to designate your program would have been replaced and so your program disappeared.

Have you ever had a picture on the High-Res graphics pages which suddenly developed a lot of dotted lines such as the example shown? The reason is that the program is using up a large amount of memory either to store itself or for data when it runs. This may be for instance because you are filling up an array in even a small program. As was discussed above the values of the numbers in the memory locations can be interpreted in various ways, depending on how the machine is being used. When a number other than zero is loaded into the area reserved for the Hi-Res pages and the graphics is being used they are interpreted as plotted points, all of which are visually meaningless. This is why you have to clear the page before using it, since there is often some of this garbage to clear up. How stupid to have the graphics located in the middle of memory, you might say. This is probably true at the moment, since almost everyone has 48k with memory being so cheap, but this wasn't always so. If you only had 16k of memory, at least you could use one page of Hi-Res graphics. Then as you progressed to 24k, another page became available. These pages were fixed by the need to balance the selling point of graphics with the machine, against the fact that many people could not afford the higher memory. There is a case for revising the ROM to take account of this, but then someone else might lose out.

There is a way to overcome the problem. It requires a few POKEs to set some switches to tell the Apple to load your Applesoft program above the Hi-Res pages, so that any additions in the way of arrays or variables happen well away from the graphics area. If you are only using HGR and not the second page :-

```
POKE 103,1
POKE 104,64
POKE 16384,0
```

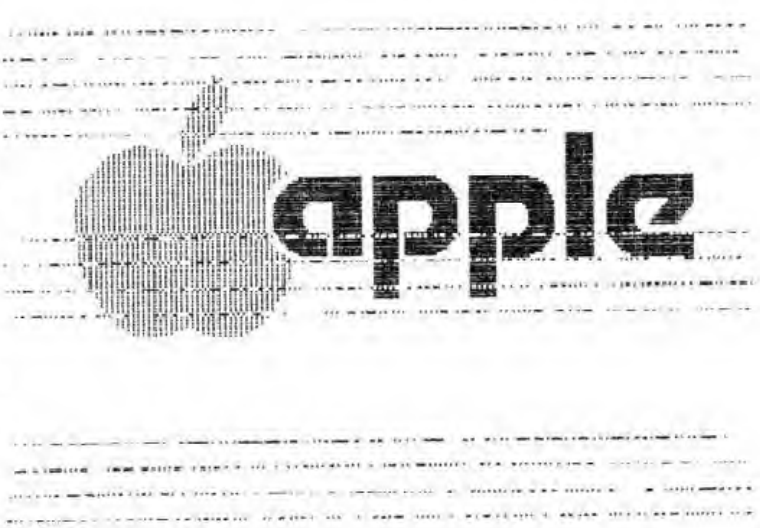
Now LOAD your program in the normal way from either tape or disk, and it will load where HGR2 normally resides.
IF YOU NOW USE HGR2 YOU WILL DELETE YOUR PROGRAM.

If you wish to be clear of both pages then :-

```
POKE 103,1
POKE 104,96
POKE 24576,0
```

and now LOAD your program as normal.

The above is not necessary with INTEGER BASIC, because the program is loaded from the top of memory down and so you would need a rather large program to encounter any trouble.



NORTH LONDON COMPUTER FAIR

By Tony Williams

For the first time in its short existence the British Apple Systems User Group exposed itself to the public eye by taking a stand at the North London Computer Fair, held at the North London Polytechnic, Holloway Road, from April 14 to 16, 1981.

Whatever the precise purpose and value of the Fair - and the people we spoke to were not really agreed on this - the chequebook-wielding crowds pounding on the doors before the official 10.a.m. opening time, the periodic shutting of the doors to counteract the fire and air pollution hazard presented by computing's very own over-population problem, the coachloads of fairgoers dumped at the entrance, the incessant loudspeaker admonitions to 'keep moving' (presumably to spread the weight more evenly over the flooring) - all this added up ample testimony that the Fair meant something to an awful lot of people.

Not that the exhibitors had anything especially new to offer. We had the by now familiar mix of hardware systems, software houses with their games, amateur hobby users clubs. The micro magazines assumed a low profile for some reason this year and while other stands were swamped three deep, they wore a desolate air. Educational applications as usual took a back seat and indeed, Tandberg apart, one had to search hard and long to find anything of real educational interest.

If, as it seems, games are to be the name of the game, excuse the expression, then the best graphics systems will call the shots. Pet and Tandy will have to place their main thrust elsewhere. Apple will still be in the lead, of course, but hard-pressed by Atari, DAI and the Acorn Atom.

Of the hardware on show I have to say that Atari-Ingersoll and the Belgian-made DAI with their more than presentable graphics and sound capabilities will give Apple a run for its money - if only because they take less of the stuff. In all honesty if one were starting afresh one would find it hard to justify buying an Apple on the strength of its graphics alone, when these other goodies are around (although we as users know that in the final analysis, Apple will come out on top on value and other counts).

The Acorn Atom set-up, just across from the BASUG stand, attracted great and unrelenting attention, because of its acceptable graphics, its up and running inexpensive Econet multi-user facility,its low start up cost, and, not least, because it is the machine chosen by the BBC for their long awaited BASIC programming course early next year. Again, could one say hand on heart to a beginner "don't choose an Acorn, you have to go for an Apple"?

Pity the harassed exhibitors on the two Sinclair stands (Sinclair and the ZX80 Users Club) for their non-stop three-day stint satisfying customers' queries and apparently doing a roaring trade taking down orders. Even the occasional Apple owner would find it tempting to fork out the fifty pounds for a Sinclair for his kids - or himself - to use as a second machine.

Why did both Microsense and Commodore, neither of whom was represented, so misread the importance and pulling power of this Fair, one wonders. Another curious observation: One amateur users club apart, there was not a TRS80 in any shape or form in sight - neither software houses, nor speciality firms had chosen it for their wares. Is the fickle finger of fate pointing at Radio Shack / Tandy?

Sharp seemed to be doing good business on the other hand, despite the high cost of their total system including disks.

Words of acquired wisdom for the future exhibitor: beware not only the disk and cassette thief but also the clued-up 10-year-old software ripoff artist who comes to the fair complete with initialized disks. After playing "Phantoms Five" he then proceeds to save it on his own disk. Sometimes he forgets to change disks. These unofficial collectors succeeded in corrupting demo disks and cassettes the length and breadth of the Fair. The DAI stand's unequal musical battle with Basug's ALF tunes ended not in a concession of defeat but because a visitor wiped off DAI's demo disk (no back-up!).

But, to return to our hard-pressed BASUG stand. What did our newly recruited members expect from the club? In first place, I was told, they wanted contact with other members to swap experience and mull over joint problems. The Contributed Software Library scheme drew some close attention, but was not the predominant reason for joining. (Various members already have many of the utilities we have on offer and a good deal else besides. It will be fascinating to see whether this shows up in a flood of contributions to the library). Few recruits seemed overly concerned at not being able to attend meetings on a regular basis for geographical reasons, and most thought membership was a good thing of itself, without being too specific. On the stand we had no need for hard sell to put the club across. The immediate response of owners out there alone with their Apples was one of relief that we would be able to put them in touch with other BASUG members living near them. It could be that providing an Apple Users Comput-A-Date service will be as useful a function of the club as establishing more regular local groups or SIG's.

A fair number of new recruits had ITT machines and were anxious to swap experiences with other owners on their particular problems - largely, as is to be expected, concerned with the graphics difficulties.

Astonishing and gratifying to see so many boys of fifteen and under wanting to use their chequebooks(!) to buy BASUG memberships (although too young yet to possess banker's cards!). Equally astonishing how many of them have their own Apples, ("No, not Dad's", they insist, "Mine!") . Incredible also in this depressed era to see customers moving from stand to stand prepared to shell out over a thousand pounds - the price is not what counts, they have to get the right system - without knowing quite what for. So many are buying for their kids. Difficult to ask them how much of this ready money comes from redundancy payments, but equally difficult to shake off the feeling that these are the farsighted ones, who want to invest their sometimes not-so-golden handshake on something more durable than a vacation in the Bahamas. Whether a newly purchased micro will justify their optimism remains to be seen.

Apart from signing up fifty odd new members at the stand, our decision to exhibit was justified on a number of counts. In particular, it showed that the group can count on the active support of a dozen or so hard core members, (hard core here meaning the small group prepared to take time off work and spend seven hours on their feet telling people what BASUG is all about!) This is a real achievement.

So our thanks go to Leo Crossfield, Jeremy Ensor, Eddie Payne, John Rodger, Neil Stephenson, Frances Teo, Ian Trackman, and of course the long-suffering committee.

VISICALC - PLUS

Two short but extremely useful articles have come in to enhance the use of VISICALC

VISILIST - A REVIEW

By John Wingate

A recent talk to BASUG on 'VISICALC', in addition to revealing the value and flexibility of the program, also pointed out a few minor deficiencies. One such drawback is the impossibility of recalling a long formula from memory.

The program VISILIST from Computer Stations Inc. gets over this difficulty and is very easy to use. The VISILIST disk is booted to display a choice from the following four options:-

1. Display on Screen.
2. Print thru Parallel Card.
3. Print thru Serial Card.
4. End.

At this stage, your VISICALC storage diskette is inserted and after selecting the appropriate option, the screen returns the message-

ENTER FILE NAME TO LIST

The contents of all grid locations are then listed and the contents may be values, labels or formulae.

The program also prints out commands and parameters in effect on the file listed. For example, /GC 9 indicates that the global column width was current on the saved file.

All in all, a very useful addendum to VISICALC. The price will tend to vary from dealer to dealer, but you should have no difficulty in obtaining a copy for around eleven pounds, plus VAT.

*VISICALC and VISILIST are trade marks.

CREATING VISICALC-READABLE FILES

By A. Graham-Bryce

Visicalc is such a convenient dynamic tool that it seems worthwhile to produce any tabular output from a program in a form that Visicalc can read, so that it can be easily manipulated. This is not too difficu;lt as they are straightforward sequential files. Essentially the file consists of table entries and format commands.

TABLE ENTRIES all start with > followed by the coordinates and : .

There are three kinds of entry:

1. values e.g. >A20:422

2. labels e.g. AD14:"TOTALS

3. formulae e.g. >113:+B4*15-6 or >H25:'SUM(H2.....H24

FORMAT COMMANDS all start with / as they are entered into Visicalc. The commands themselves are also the same e.g. /GC8 for Global column width 8 or /W1 for one window.

In order to relate the alphabet column headings in Visicalc to outputs based on a normal numerical loop in my program, I found it most convenient to put them in an array, CL$ thus:

```
10 FOR I = 1 TO 63

20 CL$(I) = CHR$(64+ I)

30 IF I > 26 THEN CL$(I) = "A" + CHR$(64 + I -26)

40 IF I > 52 then CL$(I) = "B" + chr$(64 + I - 52)

50 NEXT
```

This note is not exhaustive on the subject, but it does produce files which Visicalc can read.

PASCAL PAGE

Author: Frank Kay.

Recently received from International Apple Core is a Pascal Disk containing the following:-

- A Fortran Fixer utility, which patches Fortran code files to make them fully consistent with the Pascal Operating System. Apparently some system values are not being set in the code file by the compiler, as described by the following extract from the comments in the program:

Information in the code files which Apple Pascal 1.1 expects is not properly set by the FORTRAN compiler. USER DEFINED PASCAL INTRINSIC UNITS WILL NOT WORK WITH FORTRAN, NOR WILL BUILT-IN PASCAL INTRINSIC UNITS WORK ALL OF THE TIME WITH FORTRAN, UNLESS THE FOLLOWING POST-PROCESSOR IS APPLIED TO THE CODE FILES.

After compiling any FORTRAN unit or host program, execute the included program (FORTFIX) on the resulting code file. This will fill in the necessary missing information. This should be done for ALL FORTRAN code files even if they do not use intrinsic units or you will be running them on the 1.0 Apple Pascal system. In doing this you will be sure your FORTRAN code file will work on any Apple Pascal version.

It should be said that in the Fortran coding that I have done - nothing very advanced, but using built-in intrinsics - I haven't come across this problem.

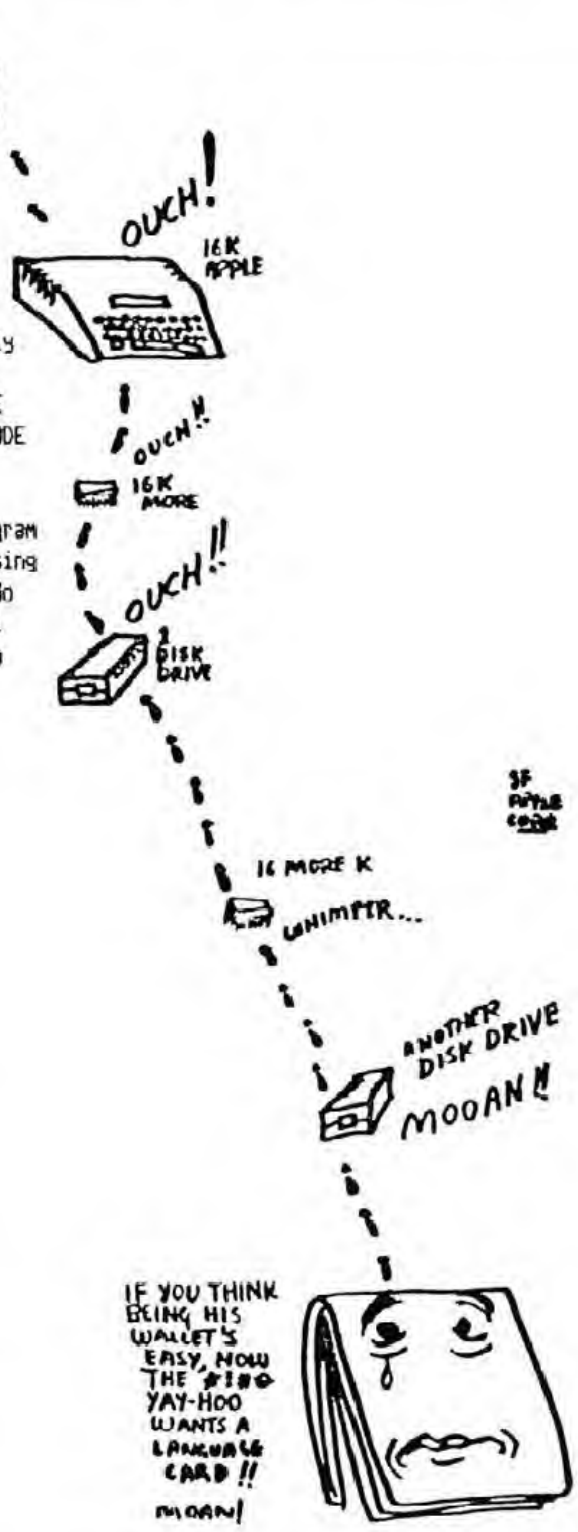
- A Listing program, which prints text files and has formatting and file-inclusion capabilities.

- A Utility and a set of documentation as described below.

Despite the fact that the documentation is dated 12-Jan-80 (!), its availability really marks the coming-of-age of Apple software. Here is a vital insight into the construction of the Apple UCSD Pascal Operating System, which makes instructive reading even if you are not concerned with interfacing new hardware with the system.

The first section describes the Utility Software and procedures for adding your own Device Driver software to Pascal in a defined and controlled manner. For those of you not familiar with Apple UCSD Pascal, the Operating System automatically recognises the presence of interface cards in the Expansion Slots of the Apple, and configures itself to use these in a defined manner. The information given in this document allows you to add suitable software to control new devices, or to modify the function of existing ones, without having to resort to patching the operating system code. This is very important if such an operation is to be carried out effectively by users who were not involved in the design and writing of the operating system in the first place. I am pleased to find such an enlightened attitude amongst those involved in building operating system software, instead of the rather secretive approach adopted by many.

The second part of the documentation describes the extensions and modifications to Pascal 1.0 which comprise Pascal 1.1. It therefore gives a useful description of the system as a whole, and illustrates, for instance, how software can be written which inhibits the operating system trapping the standard control characters (the Editor does this, for instance). It also shows that the operating system has been planned with further enhancements in mind, and is capable of running with interrupts enabled. I will report upon developments in exploring this in later Pascal Pages. The article concludes with an annotated listing of the Pascal 1.1 BIOS. All in all, a very significant and useful document.



pascal page

PRINTER INTERFACING - PART 2

By Chris Murphy

In the last article, I investigated the Parallel Interface; now, I wish to investigate Serial Interfaces, and look at the Apple High Speed Serial Card, the C.C.S. Asynchronous Serial Card and the Apple Communications Card. First let's define what the Serial Interface is and what it does.

Serial Interfaces

We saw that one limitation of the Parallel Interface is that there is a limit on the length of cable between the computer and the printer ; the longer the cable, the more distorted the signal. The solution to this problem is the serial interface, where bits are transmitted one at a time down one wire. However, this introduces a complication. The computer is translating information into bits, and sending zeros and ones to the device at the other end of the serial interface ; this in turn has to try and re-assemble the bits into characters so that it can do something with them. Obviously therefore, we need some means of telling the device at the other end which bits are to be used to make which characters.

There are two methods which are commonly used to provide a solution, synchronous and asynchronous transmission. Synchronous transmission is usually used when sending large amounts of information over telephone lines e.g. a remote terminal onto a large computer. What happens is that a pre-defined pattern of synchronisation bits is sent before the data, so that the device at the other end has the information it needs to decode the stream of bits which follow into a sensible format. This obviously requires the transmitter and receiver to work precisely in harmony.

The second method (asynchronous) adds a start bit to the start of the data and one or more stop bits to the end. Start bits are usually logical zeros and stop bits logical ones. Then an entire group of bits may be sent one at a time to the device at the receiving end ; it can easily decode and re-assemble the data bits, because there will always be an exact number of data bits between the start and stop bits. Most serial printers use this method.

Because information is being sent bit by bit, it is possible to vary the speed of transmission. This is what is usually referred to as the Baud Rate, where Baud may be taken as bits per second (including synchronisation bits or start/stop bits). Thus a 300 Baud line would be transmitting at 300 bits per second. Common Baud Rates used on printers are 300,600,1200,2400,4800,9600 and 19200. In the case of serial printers, it is vital that both the transmiting end (the interface card) and the receiving end (the printer) are working at the same Baud Rate.

Having established how the computer is sending data and how fast that data is being transmitted, the last piece of the jigsaw is "handshaking" ; in other words, both the device at the transmitting end and the device at the receiving end must be able to let the other know when it is ready to transmit or receive. (Incidentally, this is a major difference between serial and parallel interfaces on the Apple. Serial interfaces can be used to receive data as well as transmit. The Parallel Card can only send. This gives us another definition - RS232C (or CCITT V24). This is an industry standard connection using a 25-pin connector, where each pin has a common use. In its most basic form, a serial interface requires four wires. These are pin 2 (Transmit data), pin 3(Receive data), pin 7(Signal ground) and pin 20 (Data Terminal Ready). In addition, two other pins commonly used are pin 4(Request to send) and pin 5(Clear to Send). In order for the handshaking to work, one end of the transmission must be the DCE (Data Communications Equipment) - usually the computer itself - and the other end must be the DTE (Data Terminal Equipment) - in our case, the printer. As stated above, serial communication may be done over long distances via telephone lines, in which case Modems are required between the DCE and the DTE. Thus using RS232C it is possible to move from a one-way transmit only interface requiring four wires connected (for example, a keyboard) to two way traffic. Our last two definitions for this section concern types of two way traffic. These are Half Duplex, where although two way traffic is possible, it can only be in one direction at a time, and Full Duplex, where traffic can be in both directions at the same time. The interface between the Apple keyboard and the screen is Full Duplex, so that when a character is typed on the keyboard, the computer receives it and sends it to the screen.

Having looked in some detail at the principles of serial interfacing, let us now look at two of the three types of serial interface mentioned at the start. Because of copy deadlines, a consideration of the Communications Card will held over for the next article.

The Apple High Speed Serial Interface Card

The Apple High Speed Serial Interface Card (HSSIC) can handle three main functions :-

i) The HSSIC can be used to send output from the Apple to a serial printer or other serial device, and can supply line feeds after carriage returns.

ii) Input may be taken from an external device via the HSSIC.

iii) The Apple can handle Half Duplex communication at various Baud Rates with a printer, another Apple, or any RS232C device.

For the purposes of this article, we will only concern ourselves with printers.

The card comes with a standard DB25 connector, with six pins connected. Only three of thses are actually used :- pin 2 (characters received from an external device) pin 3 (characters sent to an external device) and pin 7 (signal ground). Pins 4 and 5 have been jumpered together, as have pins 6, 8 and 20. No connection need be made on these.

In addition, there is a switch block with 7 dip switches which control the way the interface works. Switches 1, 2 and 3 control the Baud Rate (available Rates are 110, 134.5, 300, 1200, 2400, 4800, 9600 and 19200. Switch 4 allows a carriage return delay to be implemented. If off, the interface will wait for about a quarter of a second after transmitting a carriage return to allow the printer to complete the movement. This is usually only needed on slow printers. Switches 5 and 6 control the output line width. Possible values are 40 chars/line (with video enabled), 72, 80 or 132 with video disabled. Switch 7 specifies whether the interface is to supply a line feed after a carriage return or not.

Using the HSSIC with Basic is very simple. You need the PR#n command to intialise the interface (where n is the slot number of the interface) ; all subsequent PRINT commands will go via the interface until a PR#0 returns output to the screen. The only real peculiarity is that the TAB command in Applesoft can cause some strange output. In order to Tab on the printer, you should use the command

POKE 36,n

where n is an integer between 0 and 255, equal to the position to be tabbed to.

So far so good - however, there is one major problem with the HSSIC, and that is that it doesn't handshake properly. Earlier, the topic of Baud Rates was raised. To relate this to printers, if a printer is capable of printing at 120 characters per second, then sending information to it at speeds in excess of 1200 Baud will require handshaking. The standard HSSIC doesn't do this. If your printer is not switched on or not on-line, the HSSIC will still carry on sending information to it (unlike the Parallel card). If your printer is running at 120 c.p.s. and transmission is being carried out at speeds of over 1200 Baud, you will certainly lose information when producing program listings. Finding the ideal Baud Rate is to a certain extent a matter of trial and error.

This situation has recently changed a little, in that the Apple HSSIC is now being supplied with two Proms, the P8 (the old standard one, fitted on the card) and the P8A, which may be fitted as an alternative. The P8A was designed specifically to provide proper handshaking with Qume, Diablo, NEC and some other daisywheel printers, so that the interface stops sending information if the printer is not ready (for example, not switched on or out of paper). Also, TAB will work correctly. Using the P8A prom will handshake correctly with some other printers (for example, the Walters BD-136) ; technically, if your printer uses an ETX/ACK protocol for signalling not ready etc., then it may well work with the modified HSSIC.

I have recently received some information on a combined hardware/software patch to enable the HSSIC with a P8 prom to provide better handshaking ; however, I haven't as yet had a chance to try it out. Certainly, if it proves useful, it will feature in a later article.

Having pointed out some of the problems with the Apple card, there is another serial interface card generally available which overcomes many of the problems. This is the California Computer Systems Asynchronous Serial card.

The C.C.S Asynchronous Serial Card

Again, this card is supplied with a female DB25 connector, and has four DIP switches for setting the Baud Rate. It provides for the following Baud Rates :-

50, 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200 or set from an external clock.

It has a number of default requirements, in that it expects to work in full duplex; it expects that the printer will supply a carriage return after a line feed ; and it does not usually bother about parity checking. It provides three handshake lines. Pin 20 on the connector (Data Terminal Ready) is used for signalling by the printer. Most serial printers will drop Data Terminal Ready (DTR) if they are not ready to receive, and the CCS card will stop sending information if DTR is not high. The other two pins are 4 (Ready To Send - RTS) and pin 5 (Clear To Send - CTS). RTS is used when the card is ready to send information to the printer, and CTS is used so that the printer can acknowledge that it is ready to receive. Because of this, the CCS card is a much more versatile performer, especially with printers with large buffers, where it is possible to send information faster than the printer can print it out. It will be loaded into the buffer, and as soon as this is full, the printer can drop CTS to indicate this to the card, which will then wait until CTS goes high again, which indicates that the printer is ready for more data.

As a slight digression, the printer which I use for program development is a DRE 8840 running at 240 c.p.s. driven at 9600 Baud through a CCS serial card, the performance of which has been compared to something proverbial off a teflon coated shovel.

For those who are interested in more complex interfacing, it is possible to replace the two 256 by 4 bit Roms on the card with Rams and write your own inerface driver routines.

Using the standard card under Basic is simply a case of issuing a PR#n (where n is the slot number of the card) prior to printing, and PR#0 to return to the screen. It is possible to modify the number of characters per line with a POKE 1528+slot number with the number of characters per line. Thus POKE 1529,220 would specify that for a printer in slot 1, the required output width is 220 characters per line.

In the next article, I shall be considering the Apple Communications Card and how to use this to drive a printer.

INSURANCE

B.A.S.U.G have been able to negotiate an all risks insurance policy for members. The annual premium is £1.5 for every £100 to be insured, less a discount of 15%. For further details and proposal forms, please contact John Rogers.

SCREEN FORMATTING

By Nik Spicer

Most computer users, even programmers, regard a program as a "black box". You type in something and a process occurs. You may think "that's clever" and pull apart the program to see how it all happens, but still it remains a "black box" during day to day use - even if you know what happens inside.

Program users expect programs to work and are often completely oblivious to any subjective notions such as "is it efficient" or "that's a slow routine".

The single most impressive aspect of any working program is the way it listens and speaks to the outside world, because it is on this that a user bases opinions about it.

Programs that ask questions and display results in an untidy way will be seen as bad programs no matter how good the routines are after pressing RETURN. So will a program that crashes during an important data gathering routine due to inputs and conditions that were not anticipated by the programmer.

The following routine is my own standard array input routine, pulled almost straight off D/DATABASE. It is very similar in concept to G.I. Clements' routine in Issue 1 of Hardcore - but it goes much further, defining screen layout, re-input of strings, forward and back cursor, up and down the screen input, etc, with full error handling.

Parts of this routine are a bit obscure in operation, so here are line by line comments.

10 Clear screen: The RESTORE ensures that the screen input format is displayed - other data statements might interfere: gosub display: gosub input routine.

190 Read # fields: Read & print each field, setting each field right justified at vertical line VT(I). LE(I) = allowed length of each input field.

300 Set field counter to 0.

310 Increment field counter by 1:Clear input string (RO$).

320 Check if field # OVER range, if so then exit subroutine.

330 Check if field # UNDER range, if so then exit subroutine.

340 Put cursor at vertical line VT(I), Horizontal column 16.

360 GET one character from keyboard: If it's RETURN and this field has characters then 460.

370 if character is RETURN (and nothing in this field) then 310 (next field).

380 If backspace and 1 or 0 characters on line then clear line.

390 If backspace and characters on line then knock off last character on line & redisplay line.

400 If ESC and characters on line then clear line.

410 If ESC and no characters on line then jump UP one line.

415 If forward cursor key (->) and a character(s) have been deleted then restore last deleted character.

417 If forward cursor key and no characters have been deleted then ignore.

420 If any other control characters then ignore.

430 If length of this input line is longer than allowed then beep and ignore.

440 Print character

450 Add character to input string (RO$) and set mask string = RO$.

460 (From RETURN with characters on line) Set final result to R$(I): Next line.

470 Dropped off bottom of input fields so RETURN.

Of course anyone looking at this routine should ask WHY? There are many complex routines that look great but actually don't do a lot for the aggravation/ memory thay take up. So note the following features:

1) Control characters are ignored.
2) You have complete control of input as a user or as a programmer.
3) You have complete control of screen format, including wrap around if input string length requires it (Sorry D/DATABASE users!)
4) Back space and forward cursor restoration is available.
5) Quick editing of errors in input including back up to previously entered fields.
6) Logical exit out of the routine, either out the top or out the bottom.
7) RETURN down the fields and ESC's up the fields do not delete created strings.
Incidentally, although the routine looks slow, it runs fast enough for any touch typist.

The routine on line 190 is also useful for two line screen index displays.

If anyone feels inclined to modify this routine, and I'm sure that a lot more can be done to it, don't keep it to yourself - send it to Hardcore!

```
10  TEXT : HOME : RESTORE : GOSUB 190: GOSUB 300: HOME
: FOR I = 1 TO NF: PRINT R$(I): NEXT : END
190  READ NF: FOR I = 1 TO NF: READ NF$,LE(I),VT(I): HT
AB 15 -  LEN (NF$): VTAB VT(I): PRINT NF$: NEXT
191  DATA 10,NAME,18,10,ADDRESS,18,11,ADDRESS,18,12,ADD
RESS,18,13,ADDRESS,18,14,POST CODE,10,15,TELEPHONE #,15
,16,RENEW DATE,8,18,PAID DATE,8,19,MEMBER #,8,21
195  RETURN
300 I = 0
310 I = I + 1:RO$ = ""
320  IF I > NF THEN 470
330  IF I < 1 THEN  RETURN
340  VTAB VT(I): HTAB 16:R$ = R$(I)
360  GET Y$: IF Y$ =  CHR$ (13) AND  LEN (RO$) THEN 460
370  IF Y$ =  CHR$ (13) THEN 310
380  IF Y$ =  CHR$ (8) AND  LEN (RO$) < 2 THEN  VTAB VT
(I): HTAB 16: CALL  - 868:RO$ = "": GOTO 360
390  IF Y$ =  CHR$ (8) THEN RO$ =  LEFT$ (RO$, LEN (RO$
) - 1): VTAB VT(I): HTAB 16: PRINT RO$;: CALL  - 868: G
OTO 360
400  IF Y$ =  CHR$ (27) AND  LEN (RO$) THEN RO$ = "": V
TAB VT(I): HTAB 16: CALL  - 868: GOTO 360
410  IF Y$ =  CHR$ (27) THEN I = I - 1:R$ = "": GOTO 33
0
415  IF Y$ =  CHR$ (21) AND (R$ = "" OR  LEN (RO$) =  L
EN (R$)) THEN 360
417  IF Y$ =  CHR$ (21) THEN RO$ = RO$ +  MID$ (R$, LEN
 (RO$) + 1,1): PRINT  MID$ (R$, LEN (RO$),1);: GOTO 360
420  IF  ASC (Y$) < 32 THEN  CALL  - 1052: GOTO 360
430  IF  LEN (RO$) + 1 > LE(I) THEN  CALL  - 1052: GOTO
 360
440  PRINT Y$;
450 RO$ = RO$ + Y$:R$ = RO$: GOTO 360
460 R$(I) = RO$:R$ = "": GOTO 310
470  RETURN

]
```

READER'S LETTERS

Dundonald,
Belfast.

Dear Mr Sharp,
Very glad to learn about the setting up of your group - the U.K. group sent me one circular letter but nothing else. I am glad to know that there is a prospect of something new for the Apple II as I have often been envious of the amount of user group activity in connection with the Pet. I hope the group will flourish and look forward to hearing from you.
Hardcore has unfortunate associations; is there any possibility of an alternative.
I notice the emphasis on programming and similar areas. For myself I am interested in what trhe machine can be made to do using general purpose programs such as data base management, text processing (this letter is 'Apple Written'), statistical suites, financial control, and so on.
Without in any way belittling the serious programmer and hobbyist these programs seem the way forward for people like myself who are fascinated by the possibilities of the micro and are anxious to experiment without getting locked into programming. But that is a bit of a hobby-horse with me.

Warm regards,
Richard L.McCorry.

xxx This is the first comment - good or bad - that we've had about the title. What do the rest of you think?

xxxxxxx

Chelmsford,

Dear Mr Perry,
I have an Apple computer with DOS. I would like to join BASUG. I am particularly interested in the contents of your journal or any other information you publish because I am very severely disabled and am trying to develop, with the aid of friends a voice operating system. This is important as I have only one weakened hand to use.

Yours sincerely,
Joyce Nixon.

xxx There is an enormous and untapped reservoir of help for the disabled in all the various applications of micro-electronics, and one of the most worthwhile results of BASUG and other user groups will be in helping to apply these developments to particular and specialised problems.

xxxxxxx

Newcastle under Lyme,
Staffs

Dear John,
... After the very poor quality of the two British magazines it has been a treat to discover the wealth of useful material which is put out by the American magazine 'Nibble'. The BASUG also promises to be another invaluable source of programs and other information.
Having recently joined the group I have been most impressed by the high quality of the introductory disk and with 'Hard Cores' 1 and 2. I agree entirely with the point made in the newsletter concerning members also trying to put in as much effort as they can into providing information for the club. I have little free time at present and so am unable to develop my own programs as fast as I would like. However, I am about a seven-finger typist and I generally type in all the programs in magazines which interest me. I enclose a disk under DOS 3.3. containing some programs which might be of interest. Unfortunately I have made a grave error and consigned my copies to the dustbin during one of my very infrequent throw-out sessions of out-of-date paperwork. As you imagine I am very anxious to obtain replacement copies...

With best wishes and apologies,
Phillip R Rowley

xxx We have heard of garbage collection before, but this is ridiculous. Replacement copies are winging their way to Member Rowley. His interesting disk contributions will gain him a useful sheaf of credit points in the Software Library.

xxxxxxx

Surbiton
Surrey

Dear John,
I enclose a further modest contribution to the library on a DOS 3.3 48k disk...Incidentally, all my programs seem to be small and most of my wants large...I am wary of getting too much into debt.
Incidentally is it possible to distinguish Integer BASIC programs that will run with the program on the Intro. disk from those that won't?
Next a couple of problems. Most machine code programs seem to start at location decimal 768 (Hex 300). Consequently you can only use one at a time. Is there a straightforward way of relocation to overcome this?
Secondly, I want to be able to use all the print options available with my printer (EPSON MX 80F/T) from within an "Applewriter" text file and I have been struggling with this problem for some weeks. Is the "Go-Between" referred to on Page 5 of Hard Core available at all and will it work with other printers?
Next, the Software Library referred to on Page 23 of Hard Core. There appears to be a number of problems here. Looking at the Catalogs of the 16 different disks my first reaction was that they need sorting into categories ... it is also very hard to select programs based on a several character title. A few sentences are required at least such as on Pages 14 and 15 of hardcore No 1. I fully appreciate that what I am suggesting involves a lot of work, but I think that it would allow members to get much more out of the library... I would certainly be prepared to become a 'software father'.
Finally, is there any chance of the 'Starting Machine Code' being repeated on a Sunday... from Surbiton a Tuesday evening is virtually impossible.

Yours sincerely,
Anthony Freedman.

xxx Mr Freedman may be modest but his contributions are not. He need stand in no fear that his creditworthiness with our Software Library will be in doubt.

xxxxxxx

Norbury,
London.

Dear David,
The question was recently asked as to why an 'out of memory' error message sometimes appears when loading a short program.
It will usually be found that this problem occurs after running a Commercial program which is in, or contains a segment of, Machine Code. Such programs often protect themselves by moving HIMEM down. In many cases they load at a location which is common to 48K, 32K and 16K machines. It follows therefore that on a 48K system there is a lot of free memory lurking above the Machine Code program. This memory is not useable by Applesoft since it is above HIMEM.
The answer to this problem is to execute an FP or a DOS re-boot before loading the BASIC program. (If using Integer Basic, use INT instead of FP.)

When I first tried the Eamon Adventures disk, I thought there was a bug in it since there was a tendency to bomb out in the beginners cave. An Undefined Statement Error was shown. I think other members may have experienced this problem also. My system is 48K and hence I ignored the program 'Set Up For 32K' on the Eamon disk. However, a friend ran that program with the result that 1. The Beginners Cave no longer bombs, and 2. 'Set Up For 32K' changed itself to 'Set Up For 48K'!!
Does anyone have any Eamon Adventures other than the Beginner's Cave? If so, how about putting them in the library?

May I, in turn, appeal for help on several counts?

1. Does anyone know how one can get the Apple speaker sound to come from the TV speaker? I presume one will need to amplify and buffer it, and then mix it suitably with the composite video into the modulator. A circuit diagram and component list for a suitable mixer/amplifier would be much appreciated.

2. Is there anyone who knows how to modify 'Sargon II' so that it will print out the list of moves made?

3. Can anyone tell me of a Hardware or Firmware difference between an Apple and an ITT 2020, which can be tested easily from software? I want to make a program identify unambiguously the type of computer in which it is being run.

4. Would anyone lend me the instructions for use of any of the following ACT Appleware programs, which I bought cheaply on a 'sold as seen' basis at the North London Fair :-
L.I.S.A; Tiny Pascal; 6502 Forth; Apple Pie; Invasion Orion or Starfleet Orion?

5. If one has two Disk Drives, one with a DOS 3.2 13sector controller and one with a DOS 3.3 16sector controller, is it possible to load programs from one and save them onto the other?

Yours sincerely,
Vernon Quaintance.

xxx There are several more EAMON adventures coming up, as soon as we sort out the distribution logistics, David Row has offered to co-ordinate an Eamon/Adventure SIG - more news next issue.

xxxxxxx

Plymouth

Dear Mr Bolton,
... I must say how impressed I was by the quality and value of the two magazines you sent me. In the few days I have possessed the material I have learned a great deal more about the Apple.... My own interests lie in the field of micros in primary schools. If this were thought to be of general interest I could put a few ideas together on the subject. Regarding software ... material I have written recently is associated with work I am engaged in with gifted ten-year-olds and uses the Apple to create a text file of data that the children have gathered regarding their friends, addresses, hobbies, etc. The idea is to demonstrate a capability of the micro to the children and then use it to analyse data. Any interest?

Yours sincerely,
Roy Garland

xxx An Education SIG (Special Interest Group) is on the point of being formed - anyone interested please write in. We hope that Roy Garland will write an article for Hardcore on the above subject.

xxxxxxx

Mons, Belgium

Dear John,
I recently received Hardcore # 2 and I enjoyed it very much. I liked your Beginner's Pages very much since they present material which is simple, almost obvious and should be used by everyone - but obviously isn't (myself included). Your column on Product News was also interesting, however, as you have probably seen by now, the DOS Board is actually just a RAM card containing a slightly modified DOS which the user loads with a special program from Computer Data Services.
For those who liked the HIRES Mystery House from On Line Systems, there is a new adventure from Highland Computer Services called Creature Venture which is also in black and white HIRES and takes place in an old mansion. I'm currently stuck about half way through it. It's not bad, but I still prefer the HIRES Mystery House. Also after solving the Mystery House the HIRES adventure # 0 from On Line Systems is almost too easy to solve. While the color graphics are excellent it is not a very challenging adventure. I would only recommend it for beginners.
For anyone who has the program Applewriter, DOS 3.3, the DOS Toolkit and the Apple Silentype printer I would recommend Computer Station's Graphic Writer program. This gives you the possibility of printing using any of the twenty odd character sets from the DOS Toolkit and also gives you true descenders and, in my opinion, more readable print. It can also be used from within a BASIC program independently of Applewriter. I enclose a disk containing documentation for the PRINT USING program which I promised you and I have included two other programs (with documentation), the last one (Apple Speed) being a slow list (stop list using Game Paddle # 1).

Sincerely yours,
Richard Sylvester

xxxxxxx

OVERCOMING AUTOSTART

By Michael Mathison

The APPLE II AUTOSTART ROM is a mixed blessing, especially to those prone to working with assembly language. The editing facilities are very good, as is the <CTRL> S, but the RESET can be irritating to the extent of nervous breakdown. Because the ultimate destination of a reset is determined by RAM any overwriting of the page three addresses which determine this can be disastrous. Usually this results in the system re- initialising, which can be painful to any BASIC programer but not too fatal to any assembly code. However, sometimes the 'Power-up Byte' gets set correctly, usualy resulting in a one way trip to limbo the next time RESET is pressed. The only way out of this is to switch off.

It is in partial answer to this problem that I developed the following technique. The only draw back is that you require a language card (or it's equivalent) to operate it.

The program creates an EXEC file under DOS 3.3 called RESET which when EXEC'ted will cause the APPLE RESET to behave just as it would on an old APPLE or an ITT:

```
10 PRINT"OPEN RESET":REM CTRL-D AFTER FIRST "
20 PRINT"WRITE RESET":REM CTRL-D AFTER FIRST "
30 PRINT"NEW"
40 PRINT"CALL-151"
50 PRINT"C089 C089"
60 PRINT"D000<D000.FFFFM"
70 PRINT"C081 C081"
80 PRINT"D000<D000.E000M"
90 PRINT"FFFC:59 FF"
100 PRINT"A5C0:80"
105 PRINT"C080"
110 PRINT"":REM CTRL-B AFTER FIRST "
120 PRINT"TEXT:CALL-936"
130 PRINT"CLOSE RESET":REM CTRL-D AFTER FIRST "
```

If this is RUN and EXEC RESET is typed then all will be as normal, except:

1) Pressing RESET will bring you to the monitor, without DOS, so you will have to 3D0G to get back.
2) Anything in the card will be removed, and you will be operating in your native ROM BASIC.
3) DOS will be unable to disconect you. The only easy wway out is to reboot.
4)the DOS will be slightly altered,so dont INIT any disks.

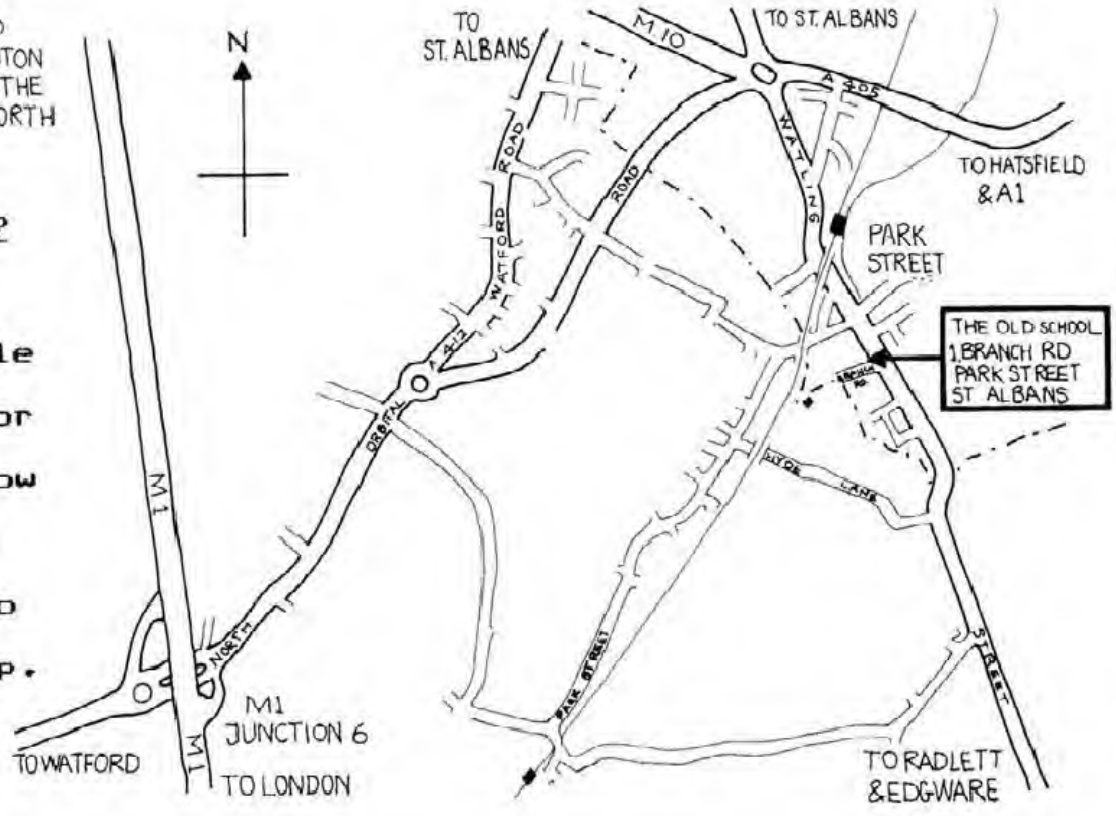
To use this with INTEGER BASIC loaded into the card simply delete lines 60,70,80,100 and 105, replacing with:

```
30 PRINT"FP"
```

On entry to the BASIC in the card the old RESET will apply.

WHERE IS PARK STREET?

Several people have asked for details of how to get to meetings, so here is a map.



SBD SOFTWARE
15 JOCELYN ROAD
RICHMOND TW9 2TJ
Tel: 01-948 0461 Telex: 22861

UTILITIES

		Retail Price
CRAE-Editor	Highlands	14.95 Disk
CRAE & MCAT -Editor & Catalog	Highlands	19.95 Disk
Applelist	SBD Software	14.95 Disk
Super Disk Copy III	Sensible Software	24.95 Disk
Multi-Disk Catalog III	Sensible Software	19.95 Disk
Apple Protector III	Sensible Software	60.00 Disk
Applesoft Program Optimizer (AOPT)	Sensible Software	19.95 Disk
Applesoft-Plus Structured Basic (APLUS)	Sensible Software	19.95 Disk
Disk Organizer	Sensible Software	19.95 Disk
Disk Recovery	Sensible Software	24.95 Disk
Quickloader	Sensible Software	17.95 Disk
DOS Plus	Sensible Software	19.95 Disk
Visicalc	Personal Software	75.00 Disk
Word Processing - MAGIC WINDOW	Artsci	79.95 Disk

(You see a full 80 columns & you don't have to buy an 80-col. card !!)

GAMES

Galactic Empire I	Broderbund	12.95 Disk
Galactic Trader II	Broderbund	12.95 Disk
Galactic Revolution III	Broderbund	12.95 Disk
Alien Rain (alias Apple Galaxian)	Broderbund	12.95 Disk
Snossie (alias Puckman)	Broderbund	12.95 Disk
Hyper Head-On	Broderbund	12.95 Disk
Galaxy Wars	Broderbund	12.95 Disk
Alien Typhoon	Broderbund	13.95 Disk
Golden Mountain	Broderbund	9.95 Disk
Oldorf's Revenge	Highlands	11.95 Disk
Tarturian	Highlands	14.95 Disk
Creature Adventure	Highlands	14.95 Disk
Hi-Res Soccer	Online	22.95 Disk
Hi-Res American Football	Online	19.95 Disk
Adventure 0 - Asteroid	Online	9.95 Disk
Adventure 1 - Mystery House	Online	12.95 Disk
Adventure 2 - The Wizard & the Princess	Online	17.95 Disk

ALL PRICES PLUS V.A.T.

POSTAGE & PACKING FREE

Method of Payment:
Access/Barclaycard/Cheque

BOOK REVIEW

COMPUTER GRAPHICS PRIMER

By Mitchell Waite.
Published by Howard W. Sams & Co. Inc., U.S.A. 1979.

Reviewed by Leo Crossfield and John Sharp.

— I —

The area of computer graphics is perhaps one of the major selling points of the Apple II.

I remember when we were trying to decide upon which machine to purchase; we covered many of the available machines in depth, but kept returning to the Apple. At that time I was cursed with three major problems; the first was the lack of money (which I hasten to add has become measurably worse since we purchased the Apple). Secondly, an entire lack of knowledge in every field that was even remotely related to computers (this seems to be slowly improving), and finally there was a lack of people within our social world that knew anything about computers.

At that time, I can remember that there appeared to be an intense lack of available software for the Apple, and we were quite concerned about this. The last year has in retrospect shown our concerns to have been laughable. Needless to say the graphics feature was one of the main elements that sold us on the Apple and since then I have found that the graphics feature is indeed one of the major selling points.

With this in mind, it makes one wonder why no one has written an elementary graphics tutor for the Apple. Well, now they have. A while back I was visiting John Sharp and noticed upon his shelf a little book called "COMPUTER GRAPHICS PRIMER" by Mitchell Waite. This book has proved to be of immense interest to me and I am sure that many of you will find it likewise.

Mitchell Waite has divided this very useful book into three main sections. Chapter One deals with the generalities of computer graphics. He explains the fast development of computers and with some very nice colour illustrations and photographs shows how everything from computer simulated landing fields to artist's mixing colours and painting can be obtained on the graphics screen. He shows what is possible today and hints at the future developments in the field of graphics. He covers the areas of industry, art, and games and lo and behold all of his examples are Apple graphics demos.

Chapter Two explains the basic concepts of computer circuits and character generation - all very nicely illustrated. Waite continues by explaining the concepts of colour generation, screen density, software and language statements. It is at this point that one realises that what Waite is suggesting are the most desirable traits on a graphics machine are already available on the Apple, with such points as easy-to-use commands and the availability of shape tables. Needless to say he explains the superiority of the Apple's graphics over its competitors.
The book gives a brief survey of all the available machines with graphics capability and includes the Apple, Pet, Bally, Atari, Compucolor and of course the good old Trash 80. Waite also reviews digital cameras and graphics printers,with hints what to look for, etc.

The first two chapters are of immense interest and very well presented, but the best is yet to come! Chapter Three, "GRAPHICS PROGRAMMING", makes a more than adequate attempt at introducing the reader to the Apple's graphics facilities, and the next seventy pages take the reader step by step through graphics plotting equations, using the Apple to formulate all the example programs - which, I might add, all work properly! He gives a progressive tutorial starting from a small Applesoft program of six lines that creates a diagonal straight line, while the next example creates a curved line. Each example increases in complexity throughout the next seventy pages culminating with the user playing with hires colour vortexes and shape tables.

A further point that adds to the nicety of this book is the simplified and clear lay out of each and every page - each example is backed with a program listing and an actual picture of the program's output on a monitor screen, and each example has a fair amount of tutorial explanation.

Mitchell Waite shows that he is a competent programmer and is more than used to the Apple II. He incorporates routines that make full use of the other Apple facilities; such as sound, colour and paddles. All in all this book is well worth purchasing.

One point that I would like to mention is that Waite does take for granted a fundamental knowledge of mathematical equations. This problem is obviously beyond the scope of this small and quite complete introduction into graphics, and is not meant as a criticism of the book.

Perhaps John Sharp would be willing to give an elementary graphics come geometry course . . let's see how others of you feel?

To close, in regard to Mitchell Waite's "COMPUTER GRAPHICS PRIMER"; if anything that I have written has sparked some small interest in you then I would suggest that this book could be well worth the few pounds that it will cost you.

Leo Crossfield

— II —

A review from someone more advanced in graphics, and with a mathematics background is difficult. It is not easy to place one's mind into the position of not knowing the answer. So who will be the best people to benefit from the book. The person who is looking to buy a computer will find it very useful if they want to use graphics and want some help in deciding which machine to buy. As Leo has outlined, there is a very comprehensive survey of the machines on the market. It is obvious from this survey, that the Apple has the edge over the other machines in most respects. It is therefore not surprising that the Apple has been chosen as the machine to illustrate how to use graphics on a micro.

Having read the first part of the book, on the technicalities of graphics, I came across many snippets of information which I either did not know or wasn't clear about. It is very clearly written and easy to understand, and the actual graphics illustrations match the clarity of the text and are attractive in themselves. It does make you jealous when you see the Evans and Sutherland pictures (theirs was the system used by NASA for designing the space shuttle), but who has the thousands of pounds necessary to have one of these machines. The only consolation is that when the Apple 20 or whatever is around at the turn of the century it might well match the resolution, colour etc., at a price the personal computer buff can afford.

The survey of the graphics micros is well presented. It might have been more detailed, but in any case is probably out of date with the rapidly changing market. It would help you make an initial decision if you were going to buy a graphics micro. The comparison of price, resolution and type of graphics will enable you to go into the shop with a short list; you need to look at the actual machine anyway.

For the Apple user the last part is a must. If you are not into graphics it will help you a lot. The examples are very well presented, although not necessarily in a logical order. The simple programs are not first, which could be off-putting. The obvious missing element is a basic grounding in mathematical language. He does assume you have some mathematical background; statements like "you will remember that the equation of ....." make this obvious. There is need for a mathematical knowledge of Cartesian Geometry to achieve anything sophisticated but by following and modifying his examples you could go quite a long way. This is where it becomes difficult to comment. Geometry is in my blood; I teach an evening class in it which is why I bought the Apple in the first place. I cannot honestly say, especially after 18 months programming mainly in graphics as well, how it would feel to come to the subject with no knowledge at all. I would think I wouldn't get too far with the equation plotting. I probably would not want to. But the latter half of this section would appeal to me. It is not as difficult to plot lines, and make them into shapes and his approach there is of much more general applicability. The drawing of shape tables is good also. I was lazy and waited until I had a program to do it for me. After all a computer is there to do the work. But if I had had this book at an earlier stage I think I would have attempted the machine code vectors, whereas the Applesoft Manual didn't make it look easy. Once you have the shapes in a table their manipulation is described well. This includes use of paddles and building up games.

To sum up, with the reservation about a missing section on introducing Cartesian Geometry, I consider this a book well worth having in your library if you want to use graphics on the Apple. If you have a maths background, and have just bought your Apple and want helping over the initial hurdles, you can't really afford to be without it. However, if you are into the graphics side, you might want to read it, but at Eight Pounds Forty Pence it is a little overpriced.

John Sharp

A WARGAMER'S VIEW OF MICROCOMPUTER GAMES.

By Robin Hood

There are quite a number of computer games on the market now, both on disk and cassette. However, the disks are fast pushing the cassettes to one side, which is not really surprising as the disk holds more information and loads in seconds. At first the majority of the games available were, and still are, in the 'Noddy' class as a friend once described them (arcade level games, if you prefer!), or the Adventure type of games which have a fantasy setting in most cases.

Arcade games like Space Invaders and ABM (the Apple version of Missile Command), are entertaining up to a point, but after a while they become boring and they certainly do not pose too much of an intellectual chalenge. Most wargamers would label them as 'fun' games, fine for a relaxing hour or so and nothing more than that.

The 'Adventure' games are divided into two types. The first of these require the player to input commands in English, words like 'look', 'search','get', go door','go tree', 'climb tree' etc,etc., and until the latter part of last year these games were without graphics. Now we have such games as Hi-res Mystery House and The Wizard and the Princess (both from On-Line Systems) with graphics. But, and I am sure there are many who will consider this heresy, I for one feel that they are nothing but computerized jigsaw puzzles. There is nothing wrong with them, if you have the time and patience to sit down with pencil and paper by uour side to help you unravel the mysteries of the program. Personally I like to see the action unfolding before me on my VDU. Even with the new 'Adventures' with graphics nothing happens, as the pictures are static. When you move to a new location all you do it change the scenery.

The second type of 'Adventure' games (including some of the better SF games) are much more like it. Rescue at Rigel (SF), Datestones of Ryn, The Temples of Apshai and Hellfire Warrior (from Automated Simulations), which are reasonable, to Wilderness Campaign, Odyssey, and the Compleat Apventure (from Synergistic Software), the latter being the best micro game on the market to date. Unfortunately most of these games are for solitaire play. The good thing about them is that you only input single letter commands and your symbol/character/party faithfully obeys said order on the screen be it simply moving from A to B or indulging in hectic combat with a bunch of nasties!

So after all that, are there any games on the market that would appeal to the veteran wargamer?. Sadly I have to report that games of this nature are a bit thin on the ground, however there are encouraging signs that this state of affairs is slowly changing. In fact one company is going out of its way to cater for wargames. That company is Strategic Simulations Inc., and I intend to take a closer look at a couple of their titles. But first I would like to take a quick look at what Avalon Hill has to offer in this area, it being the company that started boardwargaming off in the first place.

Avalon Hill have produced North Atlantic Convoy Raiders, Nukewar and Midway Campaign out of five titles, the other two being Planet Miners and The Lords of Karma. As I see it, the problem with all these games is threefold. Firstly they are all geared to solo play (apart from Planet Miners), though that in itself would not stop me buying one or more of them. Secondly, they do not utilize graphics, which really does deter me since I like to see what I am doing! Thirdly, they are only available on cassette. I do not have a cassette deck and would not want one if it were offerred me free! The sooner Avalon Hill put games with graphics on disk the better because at the moment the S.S.I. games are making their efforts look very sick indeed!

That then leaves us with S.S.I. To date they have produced Computer Bismarck, Computer Ambush, Computer Conflict, Computer Napoleonics, Computer Air Combat and Computer Quarterback. Steering well clear of the last title and dropping the word 'Computer' from the rest, I will start by saying that all these games are now available in the U.K. or at least they should be by the time you read this. As yet I do not own a copy of Conflict, Napoleonics or Air Combat (I have every intention of putting that right as soon as possible!). In fact, I have seen a short demo of the opening set-up of scenario one, called 'Rebel Force'. It is a game of modern warfare using hi-res maps and very realistic unit silhouettes of tanks, infantry and artillery field pieces; impressive graphics! Unfortunately I could not stop long enough to see much more than the set up of the attacking Federal troops who were placed in the two westernmost hex rows after the terrain had been drawn randomly on the map. Whether conflict, Napoleonics or Air Combat are any good or not remains to be seen, and, editor permitting, I shall come back to them at some later date. For now I would like to concentrate my fire on Bismarck and Ambush.

BISMARCK

The game is boxed, which is somewhat unusual. Apart from Avalon Hill this is the only other company that I know that does this. Almost everyone else in the computer industry has opted for the resealable plastic bag. For the privilege of having it all wrapped up in a box you wind up paying between £10 and £12 more than you would for the average game.

There are two versions of the games... one for the TRASH... er, sorry! one for the TRS 80 which incorporates: a computer-controlled mapboard, step-by-step computer-regulated play, hidden movement, ship vs ship combat, shadowing and an historical set-up of battleships and cruisers. For two players or solitaire, with a playing time of 1 to 3 hours. (Available on disk or cassette).

The Apple II version includes all the above plus: subs, destroyers, convoys, aircraft (land and sea based), weather, fuel restrictions and refueling at sea (for the German side only). The playing time here is from 2 to 5 hours. (Available on disk only). Inside the box you will find a rulebook written in the SPI style, two plastic laminated mapboards/turn record charts, two ship/plane data cards, two system command cards, a set-up instruction sheet, two chinagraph pencils and of course, the most important thing, a 5.25" floppy disk.

The sequence of play is as follows:

SAVE THE GAME OPTION

From the start of turn two onwards the player(s) may save the game on one of their own blank disks and come back to it at a later date, or even continue to play immediately after saving it!

SHADOW PHASE

The computer will alert the player(s) if an enemy has been spotted and may ask the player to move to a specified ship or ships. In fact in the 'teaching scenario' for new players the computer will tell the player where the Bismarck will be located at the end of the first turn during this phase. As my namesake the Hood and the Prince of Wales can both reach the square in question in turn one the Bismarck should be sunk before the start of turn two!

MOVEMENT PHASE

The British player enters his password (actually a code number), and secretly moves his units on the mapboard. As all units are displayed on the screen with a letter code and as you can stack as many units in the same square as you like the laminated maps and chinagraph pencils become essential for keeping track of your units. The reason is that the screen will only display one letter per square. There are three 'Command Modes': General commands, Mode A, Ship movement commands, Mode B and Plane movement commands, Mode C. You start in Mode A and input into the computer a two letter code command (in most cases this will send your disk off for a short spin). These commands cover such things as asking for a display of the ship positions at the beginning or end of the turn. The same for planes. You may check the status of a ship, query the units in a particular square at the beginning or end of the turn and get your search values for the start or end of the turn. Or simply go into Mode B or C to move your ships and planes.

In Mode B you go through the list of ships in alphabetical order, starting with the battleships and going on to the aircraft carriers, cruisers, destroyers, convoys and finally the subs. You are prompted to input a single letter code. A P for patrol, in which case you move onto the next ship, or an M for movement. If you choose to move you will be asked to input a compass direction or two. Mode C is for the most part identical to B except that you are now moving your aircraft.

Now, of course, it stands to reason that you could make a mistake. Not an illegal one, as the computer will not accept illegal moves. Perhaps you are not happy with the location of a particular plane or ship! In this case you can get out of Mode B or C with a simple X to get back into Mode A and from there back to B or C to run

through the list again, making any changes you wish until you are happy with your move.

Once the British player has finished his move, and with the large number of units at his disposal this is the most time-consuming part of the game, it is the turn of the German to input his codeword and make his move (or the computer in a solitaire game), with the Bismarck, Prinz Eugen, 4 Wolfpacks and 3 oilers.

FAST SHADOW PHASE

The computer will report any fast shadow possibilities, resolve them and report on the result!

COMPUTER ACTIVITY PHASE

The computer now moves the ships and planes to their new locations, and reduces the fuel and aircraft endurance factors. It then goes through a search procedure.

COMBAT PHASE

If any units have been located by search, attack possibilities are determined. Any aircraft go in first, followed by subs, then destroyers and finally battleships and cruisers.

VICTORY POINT ALLOCATION

Victory points are awarded for getting convoys into home ports (or off the map), damaging ships, sinking ships, crashed planes etc. Whatever happens, the game will end if the Bismarck is sunk and one player has at least 30 more victory points than his opponent.

CHANCE PHASE/WEATHER UPDATE

The Bismarck or the Prinz Eugen might, as actually happened, send a radio message to Berlin, in which case the British will be given the current location of the ship in question. The weather update changes the visibility levels for each row of the map.

On the strategic side the games plays well, bookkeeping is kept to a minimum but it has not been completely eliminated. On the tactical side the game does not do so well. In fact you are not given too much option. You either fight or turn away, and once you have decided to fight the computer virtually takes over complete control of the tactical side of the game. Being something of a nut when it comes to tactical games, especially naval games, I cannot help but feel somewhat disappointed in this one. A good try, but not quite good enough!

(To be continued in the next issue of HARD CORE with Robin Hood's in-depth look at Ambush, a game of street-fighting in a 1944 French village. Robin Hood - real name - is the "Friendly Woodland Outlaw" who edits "Herald", the wargamers' magazine)

THE APPLE II COLLISION COUNTER

By Michael Mathison

In Volume 1, No 2 (March 81) Richard Teed mentioned the 'collision counter' in his article on 'Differences between PALSOFT and APPLESOFT hi-res graphics'. I may be wrong, but I don't think many people know exactly what the counter is, or what it does. This seems a great shame, as no mention of it is made in any of the official APPLE literature.

One of the annoying things about the high resolution graphics routines in APPLESOFT is the apparent lack of any way of telling what is actually on the screen within a program (unlike the SCRN(X,Y) function for low resolution). This is probably not a totally bad thing as the additional code required for such a function would take up quite a lot of space in the ROM's, leaving less room for the more vital features of APPLESOFT. However the 'collision counter' goes some way to help the situation.

The high resolution graphics routines within APPLESOFT include the facility to place 'shapes' on the screen in a variety of sizes, rotations and colours. This is nothing new, but what is not so well known is that while the routines are plotting the points they also check to see whether they are plotting on top of anything!

When you DRAW a shape on the screen this is what the routines do:

1) Set collision counter to 0. Having done this it proceeds to draw the shape, but every time it plots a point it asks the following searching question:

2) Is the point I am about to plot already 'on' ? If it is, then increment the counter by one.

3) plot the point.

As you may know, the APPLE has a screen of 280 by 192 points in high resolution graphics mode. Each of these points can either be on or off, that is, either non black or black. The colour of the point is irrelevant; as long as it's glowing it's 'on'. So we could rephrase the above:

2) Is the point I am about to plot glowing ? If it is, then increment the counter by one.

So what is the use of this? Well, first we must say a little more about <u>how</u> to use it.

The collision counter is in fact mentioned once amongst the APPLE literature. On page 141 of the APPLESOFT REFERENCE MANUAL. On the third line from the bottom you will see:

$EA Collision counter for high-resolution graphics.

This tells us that the collision counter is a location in the APPLE's memory (or a single PEEK). It's location is 234 in decimal, and a PEEK(234) in BASIC will reveal it's contents.

Now imagine you draw a shape on a black background and then PEEK(234). You will get '0'. For example type in the instructions for the sample shape at the end of this article (the one detailed in the manual; page 92 onwards) and then try the following:

```
]TEXT

]HGR

]HCOLOR=3:ROT=0:SCALE=1

]DRAW 1 AT 140,95

]PRINT PEEK(234)
```

You get '0'. Now draw the shape again in exactly the same place:

```
]DRAW 1 AT 140,95

]PRINT PEEK(234)
```

You get the value 12. This is because this time every point plotted was already 'on' (as you had drawn the shape before) and so each of the twelve points of the shape caused the counter to be incremented by one.

Now try this:

```
]HGR

]HCOLOR=3:SCALE=1:ROT=0:HPLOT 140,0 TO 140,191

]DRAW 1 AT 140,95

]PRINT PEEK(234)
```

You get the value 2. This is because two of the points plotted were already 'on' because of the line that had been drawn before.

Now try this:

```
]10 HGR:HCOLOR=3:ROT=0:SCALE=1

]20 HPLOT 140,0 TO 140,191

]30 B=10

]40 HCOLOR=3:DRAW 1 AT B,95

]50 IF PEEK(234)<>0 THEN END

]60 HCOLOR=0:DRAW 1 AT B,95

]70 B=B+1:GOTO 40
```

All this program is doing is continually plotting and erasing the shape until it detects a collision.

So here we have a simple to use, but rarely known feature built in to APPLESOFT that would be invaluable in games and animations. Unfortunately this only works for DRAW and XDRAW and not for HPLOT as it would slow it down too much. Incidentally XDRAW does the opposite to DRAW; if the point is OFF it increments the counter and if it is on it ignores it.

Some further points are also worth mentioning:

1) If the SCALE factor of the shape is increased the number of points plotted will increase, so making the maximum collision count higher.

2) If the ROT is changed this may also increase the number of points plotted.

3) Any shape that plots over itself will increment the counter. This can also occur when strange values for ROT are used.

4) Drawing on a coloured background will cause interesting, but handleable results.

5) Only points which are going to be switched 'on' by the shape are considered, so if you draw a black shape on a white background the counter will not register. This also applies when drawing shapes in colours other than white, as this only illuminates some of the points of the shape.

6) As a general rule, f you are drawing your shapes in any colour other than black then PEEK(234) will be non- zero if that shape is drawn on anything non- black.

NOTE: as mentioned in Richard Teed's article this will not work for the ITT as easily.

A simple shape table to experiment with:

```
]CALL-151

*1DFC:01 00 04 00 12 3F 20 64 2D 15 36 1E 07 00

*E8:FC 1D

*<CTRL>C
```

COMBINING MACHINE LANGUAGE ROUTINES WITH APPLESOFT PROGRAMS

By Neil McFerran

The most common approach taken when using machine code routines with Applesoft programs is to BLOAD the machine code routine into a reserved block of memory. This memory may be in either of two places: page 3 ($0300 to $03CF are not used by the Apple system) or in high memory protected using the Applesoft HIMEM: command to move the string array storage down from the DOS buffer areas. In this way the machine code routine may be CALLed from the Applesoft program while remaining safe from corruption while the main program is running.

On initial entry the Applesoft interpreter stores the tokenised program starting at location $0801, with location $0800 always zero, but the program itself is completely relocatable (see 'Hard Core' vol#1, no.1, p6). Applesoft keeps a pointer in page zero as to where the program starts, this is in locations $67,$68 or 103,104 in decimal. Applesoft also keeps a pointer in page zero as to where the program ends ($AF,$B0) which is updated as you LOAD programs or enter new lines from the keyboard. When you SAVE programs it is these two pointers that determine the amount of memory to be saved.

By using these two pointers it is possible to combine both Applesoft and machine code programs so that they may be treated as a single unit without any possibility of mutual corruption. The basic idea is to have the Applesoft program in two parts, with the machine code routine sandwiched between them. The first part of all Applesoft programs structured in this way is invariant and is shown in the listing below under 'LOW'.

In this program line 10 alters the Applesoft zero page 'start of program' pointer to the beginning of the 'program proper' which will be loaded into some higher location than $0801 (the normal Applesoft start).

Line 20 ensures that the location immediately before the start of the 'program proper' will have a zero in it.

Line 30 does a CALL to the Applesoft 'RUN' routine, ie: causes execution of the 'program proper' at its higher location.(Thanks to Dave Bolton for this line).

The values of 'top', 'top-hi' & 'top-lo' should be chosen so that enough space will be left for the machine code routine. The notation here is that 'top' is a 16-bit address while 'top-hi' and 'top-lo' are the high and low bytes of this value.

Very often the Applesoft 'progam proper' may be moved to a page boundary from $0800,$0801, in this case the second POKE in line 10 may be omitted. These values must be represented in the program by constants rather than variables.

For example to move the second part of the program (the 'program proper') to location $0A01 the first part of the program should be coded as follows:-

```
10 POKE 104,10
20 POKE 2560,0
30 CALL 54630
```

You will then have from locations $0826 to $09FF for your machine code routine and its workspace - remember not to use any machine code variable storage space outside this area or you may corrupt your Applesoft program or its variables. To find out the exact beginning of memory available to you for the machine code type 'CALL -151' to get into the Monitor once you have typed in this part of the program. Then type 'AF.B0<return>' to find the 'end of program' pointer, you may start at this location ($0826 if you have typed in the program exactly as it's listed above) with your machine code.

The second part of the Applesoft program is the 'program proper' that will interact with the user. For demonstration purposes I have a simple program to print out a short message. This is shown in the second part of the listing: 'HIGH'. To input this type 'NEW<return>' and then type it in, or anything else you fancy. When you have done this save it to disk as 'HIGH', say.

Note that you may duplicate line numbers between the two halves of the program as they are completely separate entities as far as Applesoft is concerned. The last line in this program resets the page zero pointers to their normal values so that Appplesoft will behave normally with any other program (or this program itself!). I suggest that you don't put this line in as yet as it will make it easier to understand what is happening when you run the total program. Remember there's no difficulty in altering this second half of your program.

We now have two 'halves' of the Applesoft program saved as separate files, all that remains is to join them together into one 'program' with a space in the middle for the machine code routine. This is how:-

type the following:

```
FP
LOAD LOW
POKE 104,top-hi:POKE 103,top-lo
POKE top-1,0
LOAD HIGH
POKE 104,8:POKE 103,1
SAVE ALL
```

The first 'FP' is just a precaution to make sure Applesoft is in its initialised condition. If you don't have a disk then this system will work equally well with cassettes, but you'll have to work out new values for the pointers if you have Applesoft in RAM (the program normally resides at $3001 upwards, with $3000 'zero' as usual).

The second 'LOAD' moves the end-of-program pointer ($AF,$B0) to include the end of the main program, while the second 'POKE's to 104,103 have no effect on this pointer but move the 'start of program' pointer back to the beginning of 'LOW'. In this way the final 'SAVE' can be made to pick up the WHOLE program.

There is one important rule when editing an Applesoft program structured in this way: after joining the two parts together YOU MAY NOT ALTER ANYTHING IN THE FIRST PART of the program, otherwise Applesoft will move your machine code routine and also the 'high' part of your program. This occurs because the interpreter 'thinks' your Applesoft program runs all the way from $801 to wherever $AF,$B0 points to.

Altering the second half of the program is not prone to introducing this type of error as all locations from 'top' to wherever $AF,$B0 points to are solid Applesoft program and present no difficulties to the interpreter. To do this either RUN 'ALL' (without line 63990), or LOAD 'ALL' and POKE 104,'top-hi':POKE 103,'top-lo' to get into the second part of your program.

But YOU MUST REMEMBER TO 'POKE104,8:POKE103,1' BEFORE SAVING THE WHOLE PROGRAM, otherwise you will lose the first half and the 'sandwiched' machine code routine. This is not too drastic provided you have the routines stored in separate files prior to joining the whole lot together.

You may now 'BLOAD' any machine code routine into the 'blank' space you have created (provided it fits!) and when you 'SAVE' the Applesoft program it will spirit it away like magic and yet still be able to 'CALL' it as required.

This type of program structure can be quite useful when a machine code routine is too large to fit into page 3 or when you have something already in there. It also has the advantage of producing a single program module that could even INCLUDE one of the hires screens with a pre-drawn logo or suchlike....

Another use for this approach is to protect some variables by poking them into a protected area you may then 'CLEAR' for a re-run of the program, followed by a seies of peeks to recover them, eg: to save the current date while resetting all the other variables.

And next issue how to 'CHAIN' programs using this procedure!

PROGRAM LISTINGS

LOW:

```
10 POKE 104,top-hi:POKE 103,top-lo
20 POKE top-1,0
30 CALL 54630
```

HIGH:

```
10 TEXT:HOME:VTAB 5:HTAB 5
20 PRINT "HELLO THERE"
30 PRINT "THIS IS YOUR PROGRAM
WORKING."
63990 POKE 104,8:POKE 103,1
```

****

THE APPLE FOOTBALL

By Graham Rubens

One of the biggest problems for the hobby computerist is finding an application which may help pay for it all. In my own case I first met any sort of microprocessor in the autumn of 1979. I bought a Tandy TRS.80 because I had heard so much about the Silicon Chip Revolution, yet could not find an English-speaking computer man.

What were all those KBYTES and RAM'S? In fact my first real stumbling blocks were the funny characters like a letter O but with a line through them, and the word PROGRAM which kept coming up in every conversation. I had a gut reaction that anyone who could master this new technology and still speak about T.V.SCREENS and TYPEWRITER KEYBOARDS could make a lot of money.

My first machine and I were locked in the front room for 48 hours. When I emerged I knew all about Basic and Computers. It really was very easy. You just keep trying until the computer stops bleeping and saying SYNTAX ERROR. Seriously though the TANDY Manuals are the best that I have seen for newcomers. I learnt an awful lot about the first concepts from that machine.

Within a few months I was keen to upgrade my system and looked at the Idea of adding Disks and a Printer. Now that I understood enough to actually talk to the salesman I could examine the other types of machine on the market.

(continued on page 49)

I disregarded the PET because of the horrible tiny keyboard. I did not think that any serious business person could regard that machine as anything more than a toy because of those little buttons!! I also disregarded the SORCEROR because of the resident eight-track tape system which looked rather untrustworthy (bearing in mind all the car cassettes which had gone out of fashion). These reasons seem very naive now, but they were my reactions and I was a "SMALL BUSINESS MAN" just like the people I hoped to sell to.

I eventually decided to go for the BRAND NAME that I felt most people could trust, and settled for a 48K ITT.2020 with 2 Disk drives and an Anadex printer. Then I sat down and spent a few months learning all about it. Like any other addictive pastime the great thing about micros is that you learn enough within a short time to enjoy learning the rest. In my case I could use the ITT within hours. Let me just add that after 18 months I'm still learning!!!

I eventually joined a Company selling ITT's to "Small Businesses" and in the summer of 1980 I sold a system to Norwich City Football Club. From this sale a new company grew, called FIRST DIVISION COMPUTERS which is run by Norwich City F.C. to market micros to other football clubs. By retaining the ability to talk in simple terms to people who DON'T understand computers, I have found the niche which I first suspected might be there when I didn't understand all those technical terms.

First Division Computers is now installing Apple systems into the football world. I have installed a 10 Megabyte System into a large football club, and have also sold other clubs the idea of an Apple (which they have bought locally). I have written several packages for the various applications of football clubs, and am now working full time as Chief Executive of First Division Computers, that means that I am the Systems Analyst, Programmer, Installations Manager, Sales Rep, Delivery Driver, Accounts Clerk, Secretary and Sweeper.

Most of the companies about to buy micros during the coming years are run by people who do not understand computers. What they really want is a supplier who can talk about ledgers and invoices, etc., and only by keeping this in sight can we all get a financial return on our investment of time and money in our new technology.

I personally feel that job satisfaction is one of the most important features of life. Most of us spend about one quarter of our lives at work and that means fifteen years of our waking time. I for one would rather spend those years doing something I enjoy.

Pippin's Page ~~~~

Edited for younger readers by Vernon Quaintance

You will recall that in the first issue of Pippin's Page I set the problem of saving the number of games played and the number of games won whilst cancelling all other variables.

Only one reply has been received, the one from Alan Sausse which I mentioned last month. He suggested that the following lines be added to the program:

```
90 POKE 768,0 : POKE 769,0
95 PLAYED = PEEK(768) : WON = PEEK(769)

3560 POKE 768,PLAYED : POKE 769,WON
3570 RUN 95
```

This is more elegant than the methods which I had seen up to then. In these, lines 90, 95 and 3560 are as Alan proposed, but line 3570 reads:

```
3570 CLEAR : RESTORE : GOTO 95
```

I regard this as a more elegant method because it uses two commands less, and can more readily be translated to other machines which may not use the CLEAR command.

Last month I suggested to Alan that he might think of the effect on his revised program of having a machine code routine loaded at $300. This is the normal location for machine code routines. However, this is the same place as we are using for our temporary storage of scores. (Decimal 768 = Hex 300.) The way around this problem is to use two bytes of page 0 which are not used by the Monitor, Applesoft nor DOS. Suitable locations are $CE and $CF (Decimal 206 & 207).

Those of you who tried last month's short program may have thought that there was a bug in it. when run it appeared to do nothing except clear the screen. there was no prompt shown and nothing happened if one pressed the space bar, return key, escape key or Ctrl-C. Those who experimented further would have found that something happened if any lettered key was pressed but not for the numbers or punctuation. This was the result of the WAIT command in Line 150. This command is not easy to use without very careful reading of the manual, but can be quite useful at times. The normal INPUT and GET commands leave a flashing cursor on the screen whereas using WAIT does not.
Try this short program and see for youself what happens.

```
10 HOME : GR
20 POKE -16302,0
30 FOR I = 0 TO 39
40 COLOR = 1 + 15 * RND(1)
50 VLIN 0,47 AT I
60 NEXT I
70 PRINT "NEW PATTERN? ";:GET K$
80 IF K$ = "Y" GOTO 10
90 END
```

Finally, let's have some real contributions from yourselves for next issue please. Remember, it does not have to be anything fancy. Send me a listing if you prefer, rather than a disk or tape. You can send things either to my home address (as given in the first Hard Core) or via the BASUG Box Number.

Enjoy Your Programming.

SALES AND WANTS

ITEMS FOR SALE

2716 EPROMs burnt in with your code. Use on ROMPLUS board, CCS Integer Card, or (with adaptor) on the motherboard.
Your Eprom burnt in £4.00
Eprom supplied and burnt in £8.00
(Add £1 for motherboard adaptor)
David Bolton Park Street [illegible]

Centronics 730 incl Apple Interface - Best Offer
Integer Card £75.00
Phone 01 883 0874

ITT 2020 48K £580 or offers
Jeremy Ensor [illegible]

ITT 2020 48K with Integer Card and Programmers Aid No.1 £550
Richard Teed Maidstone [illegible]

Disk Software - with documentation £8 each
World of Odyssey (Synergistics)
Adventure by Bob Scurry (Rainbow)
Apple Invaders (ACT)
Space II (Eduware)
Tape Software £5 each
King (Rainbow)
Apple Invaders (Creative)
[illegible]

If this ad. doesn't convince you that **Epson** produce the **worlds' best** low cost quality printers...

- the **FREE** printout samples will!!

THE MX 80 FT/1

– both single sheet (friction) and continuous (tractor) feed & 9 wire head with true descenders

£399 + VAT

THE MX 80 FT/2

– as MX80 FT/1 and with HIGH RESOLUTION GRAPHICS

£449 + VAT

Ideal for
PET, Apple,
Sharp, TRS80, Video
Geni, RM 380Z, Nascom,
Superbrain and most Micro's

NEW

THE MX 82

– The Apple II Printer !!
9 x 9 matrix + Tractor Feed
+High Resolution Graphics

£415 + VAT

MX80 Features include:
***True Descenders** – *9 x 9 dot matrix for sharp legibility.* ***Emphasised and Double Characters** – *solid characters, a real must for word processing.* ***Dual Paper Handling** – *ideal for long print runs or single sheets with up to 3 copies.* ***Systems Flexibility** – *full range of optional interfaces and graphics ROM's.* ***International Versatility** – *internally selectable characters UK, USA, French and German.* ***Full Graphics Capability** – *64 graphics blocks and on MX80 FT/2 high resolution graphics.* ***Fast through-put** – *80 characters per second, bi-directional, guasi start line logic seeking.* ***Full Control** – *with a variety of print densities to 132 columns and top of form, horizontal and vertical tab controls.* ***Reliability** – *Precision engineered quiet operation. Full twelve months warranty.*

Call us for your local dealer or return this coupon.

THE EPSON DISTRIBUTOR

Micro Peripherals

61 New Market Square,
Basingstoke, Hampshire RG21 1HW

Tel: 0256 56468 (4 lines) Tlx: 858575

Ring Ian for up to the minute prices and technical information on a whole range of Japanese Micro Products

To: Micro Peripherals, 61 New Market Square, Basingstoke, Hants.
Please send me full details of the EPSON range including printout.
Name..... Position:.....
Company:.....
Address:.....
..... PC 5
(BLOCK CAPITALS PLEASE)